

日 本 国 特 許 庁  
JAPAN PATENT OFFICE



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2001年 1月31日

出 願 番 号

Application Number:

特願2001-024513

[ ST.10/C ]:

[ JP2001-024513 ]

出 願 人

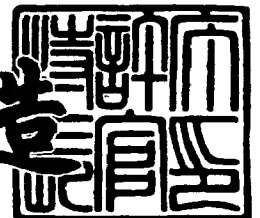
Applicant(s):

パシフィック・デザイン株式会社

2002年 1月18日

特 許 庁 長 官  
Commissioner,  
Japan Patent Office

及 川 耕 造



出証番号 出証特2001-3117389

【書類名】 特許願  
【整理番号】 000283P118  
【あて先】 特許庁長官殿  
【国際特許分類】 G06F 9/20  
【発明者】

【住所又は居所】 東京都新宿区西新宿 6 丁目 1 2 番 1 号 パシフィック・  
デザイン株式会社内

【氏名】 下郡 慎太郎

【発明者】

【住所又は居所】 東京都新宿区西新宿 6 丁目 1 2 番 1 号 パシフィック・  
デザイン株式会社内

【氏名】 鎌野 昇一

【発明者】

【住所又は居所】 東京都新宿区西新宿 6 丁目 1 2 番 1 号 パシフィック・  
デザイン株式会社内

【氏名】 北島 利明

【特許出願人】

【識別番号】 598149242

【氏名又は名称】 パシフィック・デザイン株式会社

【代理人】

【識別番号】 100102934

【弁理士】

【氏名又は名称】 今井 彰

【手数料の表示】

【予納台帳番号】 050728

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】	要約書	1
【ブルーフの要否】	要	

【書類名】 明細書

【発明の名称】 データ処理システム、データ処理装置およびその制御方法

【特許請求の範囲】

【請求項 1】 複数のデータ処理装置を有するデータ処理システムであって、2 以上の前記データ処理装置は、専用命令により実行される特定のデータ処理用のデータパス部を備えた少なくとも 1 つの専用データ処理ユニットと、汎用命令により汎用処理を実行可能な汎用データ処理ユニットと、前記専用命令および汎用命令を備えたプログラムに基づき、前記専用データ処理ユニットおよび汎用データ処理ユニットに命令を発行する命令発行ユニットとを有する第 1 のタイプのデータ処理装置であり、

少なくとも 1 つの前記第 1 のタイプのデータ処理装置の汎用データ処理ユニットは、他の前記第 1 のタイプのデータ処理装置の前記汎用データ処理ユニットとデータを交換可能な通信手段を備えているデータ処理システム。

【請求項 2】 請求項 1 において、少なくとも 1 つの前記第 1 のタイプのデータ処理装置の少なくとも 1 つの前記専用データ処理ユニットは、第 2 のタイプの前記データ処理装置とデータを交換する機能を備えているデータ処理システム。

【請求項 3】 請求項 1 において、前記第 1 のタイプのデータ処理装置は、前記プログラムを記憶するコードメモリと、前記汎用命令によりデータを入力または出力可能なデータメモリとを備えており、

前記通信手段は、前記汎用命令に基づき入力または出力するデータの入力アドレスまたは出力アドレスが予め設定されたアドレスのときに、他の前記第 1 のタイプのデータ処理装置に含まれた前記データメモリに対してデータを入力または出力するようにデータを交換するデータ処理システム。

【請求項 4】 請求項 3 において、前記通信手段は、前記出力アドレスが前記予め設定されたアドレスのときに、他の前記第 1 のタイプのデータ処理装置へデータを送信する手段を備えているデータ処理システム。

【請求項 5】 請求項 3 において、前記通信手段は、前記入力アドレスが前記予め設定されたアドレスのときに、他の前記第 1 のタイプのデータ処理装置か

らデータを受信する手段を備えているデータ処理システム。

【請求項 6】 請求項 3 において、前記通信手段は、他の前記第 1 のタイプのデータ処理装置からデータを受信すると前記データメモリの所定のアドレスにデータを記憶する手段を備えているデータ処理システム。

【請求項 7】 請求項 6 において、前記通信手段は、前記データを記憶する手段がデータを記憶する前記データメモリの受信専用領域が前記汎用データ処理ユニットにより読み出されているときは、前記データを記憶する手段の処理を延期し、前記データを記憶する手段の処理中は、前記汎用データ処理ユニットが前記受信専用領域からデータを読み出す処理を延期する調停手段を備えているデータ処理システム。

【請求項 8】 請求項 3 において、前記通信手段は、他の前記第 1 のタイプのデータ処理装置からデータを要求されると前記データメモリの所定のアドレスからデータを提供する手段を備えているデータ処理システム。

【請求項 9】 請求項 8 において、前記通信手段は、前記データを提供する手段がデータを取得する前記データメモリの送信専用領域が前記汎用データ処理ユニットにより書き込まれているときは、前記データを提供する手段の処理を延期し、前記データを提供する手段の処理中は、前記汎用データ処理ユニットが前記送信専用領域にデータを書き込む処理を延期する調停手段を備えているデータ処理システム。

【請求項 10】 請求項 1 において、複数の前記第 1 のタイプのデータ処理装置の前記専用データ処理ユニットを含む、単一のデータの流処理するデータ処理システムが形成されているデータ処理システム。

【請求項 11】 請求項 1 において、複数の前記第 1 のタイプのデータ処理装置の前記専用データ処理ユニットを含む、データの流処理する複数のデータ処理システムが形成されているデータ処理システム。

【請求項 12】 専用命令により実行される特定のデータ処理用のデータパス部を備えた少なくとも 1 つの専用データ処理ユニットと、

汎用命令により汎用処理を実行可能な汎用データ処理ユニットと、

前記専用命令および汎用命令を備えたプログラムに基づき、前記専用データ処

理ユニットおよび汎用データ処理ユニットに命令を発行する命令発行ユニットとを有するデータ処理装置であって、

前記汎用データ処理ユニットは、他の前記データ処理装置の前記汎用データ処理ユニットとデータを交換可能な通信手段を備えているデータ処理装置。

【請求項 1 3】 請求項 1 2 において、前記プログラムを記憶するコードメモリと、前記汎用命令によりデータを入力または出力可能なデータメモリとを有し、

前記通信手段は、前記汎用命令に基づき入力または出力するデータの入力アドレスまたは出力アドレスが予め設定されたアドレスのときに、他の前記データ処理装置との間でデータを交換するデータ処理装置。

【請求項 1 4】 請求項 1 3 において、前記通信手段は、前記出力アドレスが前記予め設定されたアドレスのときに、他の前記データ処理装置へデータを送信する手段を備えているデータ処理装置。

【請求項 1 5】 請求項 1 3 において、前記通信手段は、前記入力アドレスが前記予め設定されたアドレスのときに、他の前記データ処理装置からデータを受信する手段を備えているデータ処理装置。

【請求項 1 6】 請求項 1 3 において、前記通信手段は、他の前記データ処理装置からデータを受信すると前記データメモリの所定のアドレスにデータを記憶する手段を備えているデータ処理装置。

【請求項 1 7】 請求項 1 6 において、前記通信手段は、前記データを記憶する手段がデータを記憶する前記データメモリの受信専用領域が前記汎用データ処理ユニットにより読み出されているときは、前記データを記憶する手段の処理を延期し、前記データを記憶する手段の処理中は、前記汎用データ処理ユニットが前記受信専用領域からデータを読み出す処理を延期する調停手段を備えているデータ処理装置。

【請求項 1 8】 請求項 1 3 において、前記通信手段は、他の前記データ処理装置からデータを要求されると前記データメモリの所定のアドレスからデータを提供する手段を備えているデータ処理装置。

【請求項 1 9】 請求項 1 8 において、前記通信手段は、前記データを提供

する手段がデータを取得する前記データメモリの送信専用領域が前記汎用データ処理ユニットにより書き込まれているときは、前記データを提供する手段の処理を延期し、前記データを提供する手段の処理中は、前記汎用データ処理ユニットが前記送信専用領域にデータを書き込む処理を延期する調停手段を備えているデータ処理装置。

【請求項 2 0】 専用命令により実行される特定のデータ処理用のデータパス部を備えた少なくとも 1 つの専用データ処理ユニットと、汎用命令により汎用処理を実行可能な汎用データ処理ユニットと、前記専用命令および汎用命令を備えたプログラムに基づき、前記専用データ処理ユニットおよび汎用データ処理ユニットに命令を発行する命令発行ユニットと、前記プログラムを記憶するコードメモリと、前記汎用命令によりデータを入力または出力可能なデータメモリとを有するデータ処理装置の制御方法であって、

前記汎用命令に基づき入力または出力するデータの入力アドレスまたは出力アドレスが予め設定されたアドレスのときに、他の前記データ処理装置との間でデータを交換する通信工程を有するデータ処理装置の制御方法。

【請求項 2 1】 請求項 2 0 において、前記通信工程は、前記出力アドレスが前記予め設定されたアドレスのときに、他の前記データ処理装置へデータを送信する工程を備えているデータ処理装置の制御方法。

【請求項 2 2】 請求項 2 0 において、前記通信工程は、前記入力アドレスが前記予め設定されたアドレスのときに、他の前記データ処理装置からデータを受信する工程を備えているデータ処理装置の制御方法。

【請求項 2 3】 請求項 2 0 において、前記通信工程は、他の前記データ処理装置からデータを受信すると前記データメモリの所定のアドレスにデータを記憶する工程を備えているデータ処理装置の制御方法。

【請求項 2 4】 請求項 2 3 において、前記通信工程では、前記データメモリの受信専用領域が前記汎用データ処理ユニットにより読み出されているときは、前記データを記憶する工程を延期し、前記データを記憶する工程中は、前記汎用データ処理ユニットが前記受信専用領域からデータを読み出す処理を延期するデータ処理装置の制御方法。

【請求項 2 5】 請求項 2 0 において、前記通信工程は、他の前記第 1 のタイプのデータ処理装置からデータを要求されると前記データメモリの所定のアドレスからデータを提供する工程を備えているデータ処理装置の制御方法。

【請求項 2 6】 請求項 2 5 において、前記通信工程では、前記データメモリの送信専用領域が前記汎用データ処理ユニットにより書き込まれているときは、前記データを提供する工程を延期し、前記データを提供する工程中は、前記汎用データ処理ユニットが前記送信専用領域にデータを書き込む処理を延期するデータ処理装置の制御方法。

【発明の詳細な説明】

【 0 0 0 1 】

【発明の属する技術分野】

本発明は、ハードウェアで演算処理を実行可能なデータパスを備えた専用データ処理ユニットを備えたデータ処理装置およびそれを用いたデータ処理システムに関するものである。

【 0 0 0 2 】

【従来の技術】

L S I の大規模・微細化は数十年に及ぶ進展を遂げ、近年では極めて大きな機能を持ったシステムがシステム L S I などとしてシリコン上に具現化できるようになりつつある。このような背景にあって、インテル社のペンティアム L S I に代表されるような高速・高性能な汎用 L S I とは別に、目的に応じて性能を最大限に引き出す専用目的のシステム L S I や、その応用分野では汎用 L S I よりコストパフォーマンスの優れた解を引き出すシステム L S I の需要が拡大している。例えば、携帯電話に見られるような低消費電力を要求される L S I や、ネットワーク機器に見られるようなリアルタイム応答性とデータもしくはパケット転送に適した L S I、さらには、画像データの転送を目的とした画像圧縮伸長に適した L S I などの通信ネットワーク応用分野とデジタル T V に代表される情報家電応用分野で特に顕著である。

【 0 0 0 3 】

【発明が解決しようとする課題】



このような要求に対し、専用目的のシステム L S I を構築するにあたり、専用目的のプロセッサを開発および製造する方式が採用されつつある。大規模な専用システム L S I が要求される場合、システム L S I の機能、すなわち仕様は何らかの形式で高級言語（C 言語や J a v a 言語）で定義され記述される。したがって、その高級言語を実行できるコンパイラ等の環境を備えたプロセッサ、あるいはそのような環境に対応できるプロセッサであることが要求される。このため、目的に応じた専用命令を装備させた専用プロセッサであれば、その高級言語で記述された内容进行处理するための専用回路を備えているので極めてコストパフォーマンスの良いシステム L S I を提供できる。

## 【 0 0 0 4 】

一方、処理速度を向上する手法としてマルチプロセッサにより並列処理を行う手法が知られている。したがって、C 言語により記述された一つのプログラムを分割して複数の処理プロセスにし、これらの処理プロセスを並列に実行することができれば処理速度は大幅に向上する。そして、汎用プロセッサでは装備されにくい特殊な演算を処理する命令による演算処理は、汎用プロセッサではクロック数を多く費やし易いので、専用命令として専用データ処理回路で処理する設計とし、それらの専用データ処理回路により並列処理することにより大幅に処理速度が向上する。

## 【 0 0 0 5 】

しかしながら、C 言語で記述されたシステムを複数の処理あるいはプロセスに分解して、それを処理可能な専用回路が設計できたとしても、それらの処理あるいはプロセスが並列に実行されるように制御するためには、相互の専用回路の処理状況などを伝達する何らかの通信機能も専用回路に設ける必要がある。さらに、その通信結果に基づき専用回路における処理を制御する機能も盛り込む必要がある。さらに、適用分野に応じて多種多様の演算が要求されるので、それぞれの演算に対応した専用回路を開発することに加えて、専用回路同士を並列に動作させるための機能が必要である。したがって、専用回路を並列動作させるシステム L S I は、処理速度は大幅に向上すると考えられるが、そのようなシステム L S I を設計し検査するためには膨大な時間とコストが必要となる。したがって、需

要に応じてタイムリーに提供することは難しく、さらに、コストパフォーマンスも悪く、これらを改善することができる技術は開示されていない。

【 0 0 0 6 】

そこで、本発明においては、複数の専用回路を並列動作させることが可能なシステム L S I を短時間で低コストで開発し、提供することができるデータ処理システムおよびデータ処理装置を提供することを目的としている。そして、C 言語や J A V A 言語などの高級言語で記載されたプロセスを複数のプロセスに分散して並列に実行することができるシステム L S I を短時間で経済的に供給することができるデータ処理システムおよびデータ処理装置を提供することを目的としている。

【 0 0 0 7 】

さらに、ハードウェアを直接意識しないで記述できる C 言語や J A V A 言語に対応した、あるいは連動した通信機能を採用することにより、C 言語などで記述された大規模なシステムを、複数の専用回路を備えたシステムとして短時間で低コストで提供することができるデータ処理システムおよびデータ処理装置を提供することも本発明の目的としている。

【 0 0 0 8 】

【課題を解決するための手段】

本願の出願人は、たとえば、特開 2 0 0 0 - 2 0 7 2 0 2 号にカスタマイズ可能な専用命令を装着できるデータ処理装置を開示している。このデータ処理装置は、専用データ処理ユニットである V U 部と、汎用なデータ処理が可能な R I S C プロセッサに当る P U 部とを有するデータ処理装置であり、V U 部は P U 部とは異なりマルチサイクルで動作可能であり、専用命令により大規模な処理を実行可能になっている。したがって、専用命令により実行される特定のデータ処理用のデータパス部、すなわち専用回路を備えた専用データ処理ユニットと、汎用データ処理ユニットとが組み合わされたデータ処理装置を組み合わせ、汎用データ処理ユニットに他のデータ処理装置の汎用データ処理ユニットとの通信機能を搭載することにより、複数の専用回路を備えたシステムであって、それらの専用回路を並列動作させることができるデータ処理システムを短時間に、そして経済的

に提供することが可能となる。さらに、C言語あるいはJ A V A言語で記述されたシステムのプログラムファンクションそのものを専用命令として一命令化して専用データ処理ユニットで実行可能とすることにより、C言語で記述されたシステムを複数の処理あるいはプロセスに分解し、それらの処理あるいはプロセスを専用回路で高速に並列に実行することができる処理能力の高いデータ処理システムを短期間に、そして低コストで提供することができる。

## 【 0 0 0 9 】

すなわち、本発明のデータ処理システムは、複数のデータ処理装置を有し、それらのうち2以上のデータ処理装置は、専用命令により実行される特定のデータ処理用のデータパス部を備えた少なくとも1つの専用データ処理ユニットと、汎用命令により汎用処理を実行可能な汎用データ処理ユニットと、専用命令および汎用命令を備えたプログラムに基づき、専用データ処理ユニットおよび汎用データ処理ユニットに命令を発行する命令発行ユニットとを有する第1のタイプのデータ処理装置であり、少なくとも1つの第1のタイプのデータ処理装置の汎用データ処理ユニットは、他の第1のタイプのデータ処理装置の汎用データ処理ユニットとデータを交換可能な通信手段を備えている。したがって、本発明のデータ処理装置は、専用命令により実行される特定のデータ処理用のデータパス部を備えた少なくとも1つの専用データ処理ユニットと、汎用命令により汎用処理を実行可能な汎用データ処理ユニットと、専用命令および汎用命令を備えたプログラムに基づき、専用データ処理ユニットおよび汎用データ処理ユニットに命令を発行する命令発行ユニットとを有し、汎用データ処理ユニットは、他のデータ処理装置の汎用データ処理ユニットとデータを交換可能な通信手段を備えている。

## 【 0 0 1 0 】

本発明の専用データ処理ユニットは、アプリケーションなどに特化した専用回路となるデータパス部を備えており、専用命令により特化した処理あるいはプロセスを高速で実行できる。一方、汎用データ処理ユニットは、専用命令に対処する必要がなく、基本命令あるいは汎用命令を解釈して実行できる機能があればよく、汎用性を犠牲にすることなく様々なアプリケーションなどに対応した専用データ処理ユニットと共存できる。そして、専用命令および汎用命令を備えたプロ

グラムに基づき、専用データ処理ユニットおよび汎用データ処理ユニットが制御されるので、汎用データ処理ユニットにより専用データ処理ユニットを制御したり、その演算結果による汎用的な処理を行うことができる。したがって、汎用データ処理ユニットに並列処理に必要な通信手段を設けることにより、通信機能を専用回路から分離して組み込むことが可能となり、さらに通信機能をプログラムにより制御することができる。

#### 【 0 0 1 1 】

このため、複数の専用回路を有するデータ処理システムにおいて、それらの専用回路を並列に実行するために必要な通信機能を、専用回路に影響を与えず、汎用的な構成で簡単に設けることが可能であり、さらに、プログラムにより柔軟に制御できる。このため、複数の専用回路を並列実行可能なデータ処理システムの設計および開発期間を短縮でき、低コストで提供できる。さらに、通信機能はプログラムで制御が可能なので、後の変更や修正にも柔軟に対処できる。したがって、複数のデータ処理装置の専用データ処理ユニットを含む、単一のデータの流れを処理するデータ処理システムを形成したり、複数のデータ処理装置の専用データ処理ユニットを含む、データの流れを処理する複数のデータ処理システムを形成することにより、C言語やJ A V A言語などの高級言語で記載されたプロセスを複数のプロセスに分散して並列に実行できるシステム L S I として適したデータ処理システムおよびデータ処理装置を提供できる。

#### 【 0 0 1 2 】

システム全体がC言語などの高級言語で記述されており、これを複数のプロセスに分割し、各々のプロセスを本発明のデータ処理装置に割り当てる場合には、データ処理装置の間のデータ通信をいかにして行うかという課題に直面する。プロセッサ間のデータ通信はバスを通じて行う方式や、専用の通信専用ハードウェアマクロを介して行う方式が多く用いられている。したがって、本発明のデータ処理システムにおいて、データ処理装置の間の通信手段として、これらの通信専用のハードウェアを用いることも可能である。しかしながら、C言語を記述するユーザから直にデータ転送を制御もしくは管理し難いという欠点がある。すなわち、バス方式ではハードウェアであるバス自体をC言語のレベルから直接

参照するのが困難である。上述したように、C言語のような高級言語ではハードウェアを直接意識しないで記述できることに特徴があり、当然の事とも言える。また、通信専用ハードウェアマクロを使用してデータ通信を行う場合には、通信機能は専用ハードウェアが保持するのでC言語のレベルからでは細かな制御もしくはプログラミングが困難ということになる。すなわち、これらの従来多く採用されているプロセッサ間データ通信機構は、ハードウェア側の要求に基づいてボトムアップに構築されたものである。従って、C言語との連動性は要求されておらず、そのため連動性は薄いといえる。

## 【 0 0 1 3 】

しかしながら、本発明のデータ処理システムに基づき、C言語で記述された仕様に基づいてシステムLSIを設計するためには、C言語で記述されたシステム仕様からLSI化に向けてトップダウンな設計方式であることが望ましい。したがって、C言語でハードウェアを意識することなしにデータの転送が自由に行えることが望ましい。すなわち、本発明のデータ処理システムにおいては、C言語で記述されたシステム全体を複数のC言語のプロセスに分割し、それらの各々の対応した専用回路を備えた複数のデータ処理装置の集合としてシステムLSIを設計することができる。そして、複数のC言語のプロセスに分解する際に、C言語により、ハードウェアを意識すること無しにデータを転送する記述が可能となれば、複数のC言語のプロセスに分解する設計作業を円滑に進めることがはじめて可能となる。そのためには、C言語により、ハードウェアを意識すること無しにデータ転送を自由に行えるハードウェア・アーキテクチャを提供する必要がある。

## 【 0 0 1 4 】

このため、本発明においては、汎用命令に基づくデータの入出力において、そのデータを入力または出力するアドレスにより、データを他のデータ処理装置のデータメモリから入力、または他のデータ処理装置のデータメモリに対し出力するようにしている。すなわち、本発明のデータ処理装置は、プログラムを記憶するコードメモリ、たとえば、メモリのプログラム記憶領域、コードRAMあるいはコードROMと、汎用命令によりデータを入力または出力可能なデータメモリ

、たとえば、メモリのデータ記憶領域あるいはデータRAMとを有しており、通信手段は、汎用命令に基づき入力または出力するデータの入力アドレスまたは出力アドレスが予め設定されたアドレスのときに、他のデータ処理装置との間でデータを交換し、他のデータ処理装置のデータメモリからデータを入力し、またはそのデータメモリに対しデータを出力するようにしている。また、本発明の、データ処理装置の制御方法においては、汎用命令に基づき入力または出力するデータの入力アドレスまたは出力アドレスが予め設定されたアドレスのときに、他のデータ処理装置との間でデータを交換する通信工程を有する。

## 【 0 0 1 5 】

他のデータ処理装置のデータメモリに対しデータを入出力するタイプのデータ通信において、相手側となる他のデータ処理装置のデータメモリに書き込みに行くPUT型と、相手側となる他のデータメモリに読み出しにゆくGET型とを提示することができる。そして、どちらもC言語からのデータ転送を制御することができる。すなわち、PUT型のデータ処理装置の通信手段あるいは通信工程では、出力アドレスが予め設定されたアドレスのときに、他のデータ処理装置へデータを送信する。したがって、受信側となる他のデータ処理装置のデータメモリの少なくとも1部の領域を、自己のデータ処理装置のデータメモリと同じレベルで仮想的に取り扱うことができる。このため、C言語により、データの出力先を所定のアドレスにすると、他のデータ処理装置のデータメモリにデータを書き込むことができる。

## 【 0 0 1 6 】

一方、PUT型のデータ処理装置の相手となる受信側のデータ処理装置の通信手段あるいは通信工程では、発信側の他のデータ処理装置からデータを受信するとデータメモリの所定のアドレスにデータを記憶する。これにより、受信したデータを自己のデータメモリに記憶できる。したがって、C言語によりデータが書き込まれたアドレスのデータを読み込むことにより、汎用データ処理ユニットではそのデータを使用できる。この結果、C言語により発信側と受信側のデータ処理装置間でデータを転送する処理を操作できたことになる。

## 【 0 0 1 7 】

たとえば、あるアドレスを予め設定しておき、そのアドレス以上であれば、他のデータ処理装置のデータメモリへ書き込み、そのアドレス以下であれば自身のデータメモリへ書き込む、という制御を行う。この制御を行う為に、通信相手となるデータ処理装置の情報を格納するレジスタを設け、そこに送出先のデータ処理装置の識別情報、そのデータ処理装置に対しデータ転送を開始するアドレス、転送を終了するアドレスなどの情報を格納しておくことができる。

## 【 0 0 1 8 】

同様に、GET型のデータ処理装置の通信手段あるいは通信工程では、入力アドレスが予め設定されたアドレスのときに、他のデータ処理装置からデータを受信する。したがって、送信側となる他のデータ処理装置のデータメモリの少なくとも1部の領域を、自己のデータ処理装置のデータメモリと同じレベルで仮想的に取り扱うことができる。このため、C言語により、データの入力元を所定のアドレスにすると、他のデータ処理装置のデータメモリからデータを読み込むことができる。

## 【 0 0 1 9 】

一方、GET型のデータ処理装置の相手となる送信側のデータ処理装置の通信手段あるいは通信工程では、受信側となる他のデータ処理装置からデータを要求されるとデータメモリの所定のアドレスからデータを提供する。これにより、C言語によりデータをデータメモリの所定のアドレスに書き込むことにより、受信側のデータ処理装置にデータを転送できたことになる。このようにGET型においても、C言語により発信側と受信側のデータ処理装置間でデータを転送する処理を操作できたことになる。

## 【 0 0 2 0 】

データが誤り無く転送されるには、転送するデータを入力または出力する領域に、送信側および受信側のデータ処理装置が同時に入力または出力しないようにすることが望ましい。本発明のデータ処理装置は、データを転送するタイミングをプログラムで制御することができるので、そのような事態が発生しないように送信側および受信側のデータ処理装置のプログラムを作成でき、C言語により制御することができる。また、通信手段に、データを記憶する手段がデータを記憶

するデータメモリの受信専用領域が汎用データ処理ユニットにより読み出されているときは、データを記憶する手段の処理を延期し、データを記憶する手段の処理中は、汎用データ処理ユニットが受信専用領域からデータを読み出す処理を延期する調停手段、または、データを提供する手段がデータを取得するデータメモリの送信専用領域が汎用データ処理ユニットにより書き込まれているときは、データを提供する手段の処理を延期し、データを提供する手段の処理中は、汎用データ処理ユニットが送信専用領域にデータを書き込む処理を延期する調停手段を設けても良い。また、本発明のデータ処理装置の制御方法の通信工程で調停手段と同様の制御を行うようにしても良い。

## 【 0 0 2 1 】

このように、本発明は、専用データ処理ユニットと、通信手段を備えた汎用データ処理ユニットとを有するデータ処理装置を複数有するデータ処理システムを提供するものであり、本発明のデータ処理システムにより、複数の専用回路を並列実行することができるシステム L S I を極めて短期間に、そして低コストで提供することができる。さらに、本発明においては、専用回路を備えた分散処理システムであるデータ処理装置の間の通信機構を、C 言語あるいは J A V A 言語などの高級言語と連動性および対応性のあるハードウェアで実現するアーキテクチャを提供しており、1 のプロセスから他のプロセスへデータ転送が C 言語で記述でき、その結果、C 言語によるプロセスを複数のプロセスに分割が容易となり、分散処理システムの設計が可能となる。したがって、C 言語で記述された仕様を実現し、高速で処理可能な複数の専用回路を用いた分散処理タイプのシステム L S I をさらに短期間に、経済的に設計し供給することができる。

## 【 0 0 2 2 】

## 【発明の実施の形態】

以下に図面を参照しながら本発明についてさらに説明する。図 1 に、特定の処理に特化した専用データ処理ユニット（専用命令実行ユニット、以降では V U）1 と、汎用的な構成の汎用データ処理ユニット（汎用命令実行ユニットあるいはプロセスユニット、以降では P U）2 とを備えた本発明のデータ処理装置 1 0 の概要を説明する。このデータ処理装置 1 0 は、専用回路を備えたプログラマブル



なプロセッサであり、このため、実行形式の制御プログラム（プログラムコード、マイクロプログラムコード）4 a を内蔵したコードRAM 4 から命令をフェッチし、専用データ処理ユニット1 および汎用データ処理ユニット2 にデコードされた制御信号を提供するフェッチユニット5 を備えている。本例においては、このフェッチユニットFU 5 が命令発行ユニットに該当する。

## 【 0 0 2 3 】

このフェッチユニット5 は、前の命令あるいはステートレジスタ6 の状態、割り込み信号 $\phi i$  などによって決まる所定のコードRAM 4 の所定のアドレスから命令をフェッチするフェッチ部7 と、フェッチされた専用命令あるいは汎用命令（一般命令）をデコードするデコード部8 とを備えている。デコード部8 は、専用命令をデコードした制御信号（デコードド・コントロール・シグナル；Decoded Control Signal） $\phi v$  および汎用命令をデコードした制御信号（デコードド・コントロール・シグナル；Decoded Control Signal） $\phi p$  を、専用データ処理ユニットVU 1 および汎用データ処理ユニットPU 2 にそれぞれ供給する。さらに、PU 2 からは実行状態を示すステータス信号（Exec unit Status Signal） $\phi s$  が返され、PU 2 およびVU 1 の状態がステートレジスタ（状態レジスタ）6 に反映されるようになっている。

## 【 0 0 2 4 】

本例のPU 2 は、汎用レジスタ、フラグレジスタおよび演算ユニット（ALU）などから構成される汎用性の高い実行ユニット11 と、他のPU 2 との間でデータを交換する機能を備えた通信ユニット12 とを備えており、データRAM 15 を一時的な記憶領域としてデータを入出力しながら汎用処理を実行できるようになっている。これらのフェッチユニットFU 5、汎用データ処理ユニットPU 2、コードRAM 4、データRAM 15 を有する構成は、個々の機能は異なるが一般的なプロセッサユニットと類似の構成となる。したがって、FU 5、PU 2、コードRAM 4 およびデータRAM 15 を有する構成をプロセッサユニット3 と称することも可能であり、プロセッサユニット（PUX）3 からVU 1 を制御するような概念で本例のデータ処理装置10 を構成あるいは設計することができる。

## 【 0 0 2 5 】

F U 5 からの専用命令  $\phi v$  を実行する専用データ処理ユニット V U 1 は、F U 5 が供給する命令が V 命令  $\phi v$  であるかなどをデコードするユニット 2 2 と、予め特定のデータ処理を行うように制御信号をハードウェア的に出力するシーケンサ (F S M (Finite State Machine)、ファイナイトステートマシン) 2 1 と、このシーケンサ 2 1 からの制御信号に従って特定のデータ処理を行うようにデザインされたデータパス部 2 0 を備えている。また、V U 1 は、P U 2 からアクセス可能なレジスタ 2 3 を備えており、データパス部 2 0 の処理に必要なデータをインターフェイスレジスタ 2 3 を介して P U 2 で制御したり、V U 1 の内部状態をレジスタ 2 3 を介して P U 2 で参照できるようになっている。また、データパス部 2 0 で処理された結果は P U 2 に供給され、P U 2 ではその結果を利用した処理が行われる。

## 【 0 0 2 6 】

本例のデータ処理装置 1 0 は、コード R A M 4 に、汎用命令 (P 命令) および専用命令 (V 命令) を含んだプログラムが記憶されており、それがフェッチユニット 5 でフェッチされ、デコードされた制御信号  $\phi p$  または  $\phi v$  として V U 1 および P U 2 に供給される。V U 1 は、制御信号  $\phi p$  および  $\phi v$  のうち、自己を起動する専用命令の制御信号  $\phi v$  が供給されると稼動する。一方、P U 2 には、汎用命令がデコードされた制御信号  $\phi p$  だけが供給されるようになっており、V 命令をデコードした制御信号  $\phi v$  は P U 2 には発行されず、その代わりに、実行を伴わない n o p 命令を示す制御信号が発行され、P U 2 の処理はスキップされる。V U 1 は、アプリケーションなどによって変更されるものであり、V U 1 に指示を出す専用命令もアプリケーションによって変わることが多い。V U 1 は、アプリケーションに特化した専用回路であり、V 命令をデコードした制御信号を解釈するように設計することは容易である。一方、P U 2 は、n o p 命令が出力されることにより、V U 1 用に特化した命令に対処する必要がなく、基本命令あるいは汎用命令を解釈して実行できる機能があればよく、汎用性を犠牲にすることなく様々なアプリケーションに対応した V U 1 と共存し、これらを制御したり、その演算結果を用いて処理を行うことができる。

## 【 0 0 2 7 】

このように、図 1 に示したデータ処理装置 1 0 は、リアルタイム応答などの特殊な演算が要求される処理を実現できる専用回路を備えた V U 1 と、汎用性がある P U 2 とを有するものであり、以降においては V U P U と称することにする。この V U P U 1 0 は、リアルタイム応答性を犠牲にすることなく、設計および開発期間を短縮でき、さらに、その後の変更や修正にも柔軟に対処できるものである。また、V U 1 は、1 つに限定されることはなく、アプリケーションで要求される専用処理を処理できるように複数の V U 1 を用意し、それぞれの V U 1 を稼動する複数の専用命令をプログラムコードに含めることが可能である。さらに、本例の V U 1 は、特殊な演算処理だけでなく、プログラム中の特定のプログラムファンクションを専用回路化してプログラムを効率良く可動させることができる。そして、本例の P U 2 は、他の P U 2 とデータを交換することができる通信ユニット 1 2 を備えており、他の V U P U 1 0 と通信することにより、複数の V U P U 1 0 の V U 1 を並列に稼動させることができる。したがって、V U P U 1 0 を複数備えたデータ処理システムは適応可能な範囲が非常に広いアーキテクチャである。

## 【 0 0 2 8 】

たとえば、図 2 に示すように、C 言語により記述されたプロセスが図 2 のように、親となるプロセス C 1 と、そのプロセス C 1 からデータが転送され、そのデータに基づく処理結果を返すプロセス C 2 および C 3 により構成されている場合、これらのプロセス C 1、C 2 および C 3 を図 3 に示すように 3 つの V U P U 1 0 に割り振ることが可能である。そして、V U P U 1 0 であれば、特殊な演算処理だけでなく、プログラム中の特定のプロセスあるいはプログラムファンクションを専用回路化してプログラムを効率良く可動させることができるので、処理速度を向上できる。さらに、V U P U 1 0 は、P U 2 が通信機能を備えているので、図 4 に示すように、親となるプロセス C 1 が割り振られた V U 1、すなわち V U ( C 1 ) を備えた V U P U 1 0 から、子供となるプロセス C 2 が割り振られた V U ( C 2 ) を備えた V U P U 1 0 に対しデータが転送されることにより、V U ( C 2 ) が V U ( C 1 ) と並列に処理を開始することができる。そして、V U (

C 2) の処理結果を V U ( C 1 ) に返すことにより V U ( C 1 ) ではその処理結果に基づく処理を実行することができる。

## 【 0 0 2 9 】

同様に、プロセス C 3 が割り当てられた V U ( C 3 ) を備えた V U P U 1 0 に対しデータが転送されることにより、V U ( C 3 ) が V U ( C 1 ) と並列に処理を開始することができる。さらに、V U ( C 2 ) と V U ( C 3 ) とが並列に処理を行うことができるプロセスであればさらに並列度を上げることが可能であり、処理速度を向上することができる。このように、各 V U P U 1 0 がある時刻でひとつしか動作しなければ、非並列であり、元の C 言語で記述されたプロセスを専用回路化した効果しか得られない。これに対し、本発明の V U P U 1 0 であれば、専用回路化した複数の複数のプロセスを並列に実行することが可能となり、処理速度を大幅に向上できる。このため、図 3 に示すように、C 言語で記述された仕様を複数のプロセスに分け、複数の V U P U 1 0 の V U に割り当て、それらの V U P U 1 0 によってシステム L S I 3 0 などのデータ処理システムを構築することにより、プロセスあるいはファンクションを専用回路化するメリットに加えて、それらの専用回路を並列に実行することが可能となる。したがって、非常に処理速度の速いシステム L S I 3 0 を提供することが可能となる。

## 【 0 0 3 0 】

すなわち、図 5 に示すように、C 言語で記述されたある仕様 5 1 が与えられたときに、その仕様 5 1 をある程度並列実行可能な複数のプロセス 5 2 に分けることが可能である。そして、専用回路を形成するデータパス部 2 0 とシーケンサ 2 1 により、プロセス 5 2 の全てあるいは一部が実行できるように V U 1 を生成することが可能であり、V U P U 1 0 として供給することができる。そして、そのようにして作成された V U P U 1 0 を組み合わせてシステム L S I 3 0 とすることにより、並列度の高い処理が可能なシステム L S I 3 0 を提供することができる。さらに、V U P U 1 0 は、専用回路で処理することが適さない処理はプロセッサとしての機能を備えた P U 2 により処理することが可能であり、専用回路による処理を並列に実行できるのみならず、汎用プロセッサによる処理も並列に実行させることができる。

## 【 0 0 3 1 】

図 6 ないし図 8 は、通信機能を備えた本発明の V U P U 1 0 によりデータ処理システム 3 0 を構成する幾つかの例を示してある。多くのケースでは、1 つのチップに複数の V U P U 1 0 が搭載され、本明細書に示したような構成のデータ処理システム 3 0 は、特定のアプリケーションの処理を効率良く行うことができるシステム L S I として提供されることになるであろう。図 6 に示したデータ処理システム 3 0 は、V U P U 1 0 の P U 2 と通信するのに適したアーキテクチャを備えたプロセッサ 3 1 を中心に、複数の V U P U 1 0 が適当な通信手段により接続されているものである。たとえば、並列に稼動する複数の V U 1 により、画像データとなるビットストリーム 3 9 に対し圧縮あるいは解凍に必要な一連の処理を順次施すことが可能であり、画像処理を高速に実行することができる。そして、各処理を行う V U 1 は P U 2 で制御され、その P U 2 は他の P U 2 とデータを交換できるようになっているので、処理の同期、調停あるいはエラーなどを適切に処理することができる。これらの V U P U 1 0 は、各々が独立したプログラムコードにより動作するので、マルチインストラクションによりシングルデータフローを処理することができるデータ処理システム 3 0 を提供することができる。

## 【 0 0 3 2 】

図 7 に示したデータ処理システム 3 0 は、汎用のバスなどを介してデータを送受信することができる通信機能を備えた V U ( C O M ) を搭載した V U P U 1 0 A をインターフェイスとして用い、V U P U 1 0 を繋げたシステムと、V U P U とは異なるアーキテクチャの従来あるいは他のタイプのプロセッサ 3 2 とにより構築されている。また、図 8 に示したデータ処理システム 3 0 は、V U ( C O M ) とプロセスを搭載した V U ( C 1 ) あるいは V U ( C 2 ) の 2 つの V U を搭載した V U P U 1 0 B をインターフェイスとして、他のタイプのプロセッサ 3 2 を含めてシステムを構築した例である。このように、通信機能を備えた P U 2 を採用することにより、複数の V U P U 1 0 を用いたシステムを非常にフレキシブルに構築することが可能であり、様々な仕様のアプリケーションに対する最適な構成のシステム L S I を提供できる。

## 【 0 0 3 3 】

このように、複数のVUPU10を並列実行させることにより極めて処理速度の速いシステムLSIを提供することが可能となる。そのためには、図9に示すようにC言語で記述された機能あるいは仕様51を複数のプロセス52に分解してVUPU10を作成する必要があるが、VUPU10の間のデータ通信をいかにして行うかという課題に直面する。プロセッサ間のデータ通信はバスを通じて行う方式や、専用の通信専用ハードウェアマクロを介して行う方式が多く用いられており、本例のデータ処理システム30にも適用できる。

#### 【0034】

しかしながら、たとえば、バス方式ではハードウェアであるバス自体をC言語のレベルから直接参照するのが困難であり、C言語により複数のプロセス52に分解したときに通信機能をC言語のレベルからでは細かな制御ができない。したがって、上述したような複数のVUPU10を備えたデータ処理システムを短期間に低コストで開発するためには、C言語でハードウェアを意識することなしにデータの転送が自由に行えることが望ましい。すなわち、複数のC言語のプロセスに分解する際に、C言語により、ハードウェアを意識すること無しにデータを転送する記述が可能となれば、複数のC言語のプロセスに分解する設計作業を円滑に進めることがはじめて可能となる。そして、C言語のレベルで分割されたプロセスに基づき、専用回路化できる部分をRTLに変換して専用回路を設計および製造し、専用回路を稼動する専用命令とその他の汎用処理を行う汎用命令を備えたプログラムコードを作成し、さらにこれらをテストして完成するステップ53の負荷を軽減することができる。

#### 【0035】

このため、本例ではC言語により、ハードウェアを意識すること無しにデータ転送を自由に行えるハードウェア・アーキテクチャを通信機能として採用している。この方式の通信機能は、C言語に限定されるものではなく、より分散および並列記述の容易なJAVA言語、あるいはその他の高級言語で記載された仕様をシステムLSIなどのデータ処理システムとして実現するためにも好適である。プロセスを複数のプロセスに分散して並列に実行できるシステムLSIとして適したデータ処理システムおよびデータ処理装置を提供できる。

## 【 0 0 3 6 】

図 1 0 に、本例の V U P U 1 0 の構成を P U 2 を中心に示してある。P U 2 は、図 1 に基づき説明したように、コード R A M 4 に格納されたプログラム 4 a の汎用命令をデコードした制御信号  $\phi_p$  を実行する実行ユニット 1 1 と、通信機能を備えた通信ユニット 1 2 とを備えている。そして、本例の通信ユニット 1 2 は、実行ユニット 1 1 がデータ R A M 1 5 にアクセスするために出力するアドレス A O が予め設定された範囲のアドレスであるときは、通常の R D / W R データ R A M 1 5 N とは異なる受信用データ R A M 1 5 X あるいは送信用データ R A M 1 5 Y に対し入出力動作を行う。そして、自己の受信用データ R A M 1 5 X に書き込まれたデータを読み込んだり、他の V U P U の送信用データ R A M 1 5 Y からデータを取得することにより、他の V U P U との間でデータを行う。すなわち、本例の V U P U 1 0 のプロセッサ P U X 3 は、コード R A M 4 と、データ R A M 1 5 とが異なる、いわゆるハーバードアーキテクチャと称されるタイプである。そして、データ R A M の一部を他の V U P U 1 0 と共用したり、他の V U P U 1 0 と共用のデータ R A M を設けることにより、入出力アドレスにより他の V U P U 1 0 に対しデータ転送することができる。したがって、入出力アドレスを C 言語で記述することにより V U P U 1 0 の間の通信を制御することができる。

## 【 0 0 3 7 】

この通信方式は、通信相手の V U P U 1 0 の受信 R A M 1 0 X に出力データを書き込む P U T 型と、通信相手の V U P U 1 0 の送信 R A M 1 0 Y から入力データを取得する G E T 型に大きく分かれる。図 1 0 に示した V U P U 1 0 は、P U T 型の例である。したがって、V U P U 1 0 は、入出力可能な通常の R D / W R データ R A M 1 5 N に加え、自己の実行ユニット 1 1 に対してはリードオンリとなる受信 R A M (受信データ R A M) 1 5 X を備えている。また、通信ユニット 1 2 は、出力データ D O を他の V U P U 1 0 に送信する送信インターフェイス 1 3 と、他の V U P U 1 0 から受信した入力データ D I を受信 R A M 1 5 X に書き込む受信インターフェイス 1 4 を備えている。

## 【 0 0 3 8 】

送信インターフェイス 1 3 は、送信制御部 1 3 C を備えており、実行ユニット

1 1 がプログラム 4 a にしたがってデータを書き込む際に出力するアドレス A O があるアドレス以上であると、送信バッファ 1 3 B を経由して他の V U P U のデータ R A M (受信 R A M) に書き込む。したがって、プログラム 4 a からみると、自身の V U P U 1 0 に実体のあるデータ R A M に書き込むのと同じ操作で、実体の無い送信用のデータ R A M 1 5 Z にデータを転送することができる。そして、その実体のない送信用のデータ R A M 1 5 Z は、通信相手の V U P U に存在する送信専用のライトオンのデータ R A M 1 5 X であり、通信相手の実行ユニット 1 1 にとってはリードオンの受信専用のデータ R A M となる。

## 【 0 0 3 9 】

受信インターフェイス 1 4 は、受信制御部 1 4 C を備えており、他の V U P U 1 0 から受信した入力データ D I (送信元においては出力データ D O) を受信 R A M 1 5 X に書き込む。送信制御部 1 3 C および受信制御部 1 4 C は、それぞれ、コンフィグレーション・レジスタ 1 3 R および 1 4 R を備えている。送信用のコンフィグレーション・レジスタ 1 3 R には、送信先の V U P U の識別情報 (I D)、送信開始アドレス、転送サイズ、さらには送信終了アドレスなどの実行ユニット 1 1 から出力されるデータを転送先に送信するために必要なデータが格納される。受信用のコンフィグレーション・レジスタ 1 4 R には、受信源となる送信元の V U P U の I D、受信開始アドレスおよび受信終了アドレスなどのデータを受信するために必要なデータが格納される。また、送信元の実体の無い送信専用のデータ R A M 1 5 Z のアドレスと、送信先の受信専用のデータ R A M 1 5 X の受信アドレスが一致しない場合には、それらのアドレスの対応表を送信側あるいは受信側のコンフィグレーション・レジスタ 1 3 R あるいは 1 4 R に登録しておき、送信時あるいは受信時にアドレス変換することができる。

## 【 0 0 4 0 】

これらの送信用のコンフィグレーション・レジスタ 1 3 R および受信用のコンフィグレーション 1 4 R の内容は、例えば、P U 2 の汎用レジスタ 1 1 R を介してプログラム 4 a を通じて設定することができる。したがって、C 言語により送信および受信を行うことになる入力および出力アドレスや、アドレス変換などの初期条件を設定することができる。



## 【 0 0 4 1 】

また、実行ユニット 1 1 に入力されるデータ D I は、受信用のコンフィグレーション・レジスタ 1 4 R に格納されているアドレスの内容から、受信専用のデータ R A M 1 5 X からの読出か、通常のデータ R A M 1 5 N からの読出かを判断することが可能である。このため、受信 R A M 1 5 X の出力 D O と、R D / W R データ R A M 1 5 N の出力 D O は、受信制御回路からの信号により制御されるセレクタ 1 6 を経由して実行ユニット 1 1 の D I に供給される。この結果、プログラム 4 a は、自己が入出力可能なデータ R A M 1 5 N のデータか、通信相手が書き込んだ受信 R A M 1 5 X のデータかはアドレスの違いで制御することが可能であり、その他の操作は全く同等に処理できることとなる。

## 【 0 0 4 2 】

さらに、送信インターフェイス 1 3 は、調停回路 1 3 A を備えており、データ書込状態を示す信号  $\phi p u t$  を送出する。それと共に、送信を開始するに当たっては、相手がデータを読出中でない事を確認する必要がある、送信相手の V U P U における受信 R A M 1 5 X のデータ読出状態を示す信号  $\phi b u s y$  によって識別する。したがって、データ読出状態を示す信号  $\phi b u s y$  は転送する相手のプロセッサの数分 ( I D 分) だけ必要となる。受信インターフェイス 1 4 も調停回路 1 4 A を備えており、受信専用のデータ R A M 1 5 X から読出中は、他の V U P U からの出力データを受信できないようになっている。そのため、データ書込状態を示す信号  $\phi p u t$  を受信したときに、受信 R A M 1 5 X が読み出し中であると、読出状態を示す信号  $\phi b u s y$  を出力する。送信インターフェイス 1 3 および受信インターフェイス 1 4 で取り扱われるこれらの書込状態を示す信号  $\phi p u t$  と読出状態を示す信号  $\phi b u s y$  は、方向は逆だが同じ種類に信号である。そして、レベル信号で送出されるのが一般的である。

## 【 0 0 4 3 】

本例の受信専用のデータ R A M 1 5 X は、デュアルポートデータ R A M であるが、シングルポートデータ R A M により構成することが可能である。デュアルポートデータ R A M であれば受信しながらの読出操作が可能となり、並列性が向上し、さらに、上記のような調停回路を省くことができる可能性がある。しかしな

がら、書込みアドレス A I と、読出しアドレス R A I が同一の場合があることを考慮すると上記の調停回路 1 3 A および 1 4 A と状態信号  $\phi p u t$  および  $\phi b u s y$  を設けておくことが望ましい。調停回路を省いた場合でも、書込みアドレス A I と、読出しアドレス R A I が同一の場合を考慮すると、入力データ D I から読出しデータ R D I に RAM をバイパスしてデータを出力可能な論理回路が必要となる。

## 【 0 0 4 4 】

以上に示す送信・受信機構全体を I V C 機構 ( I n t e r V U P U C o m m u n i c a t i o n 機構) と称することとする。

## 【 0 0 4 5 】

図 1 1 に、I V C 機構を備えた 2 つの V U P U 1 0 の間でデータ交換される様子を各々の P U のメモリマップ 1 9 を用いて示してある。本図から分かるように、P U T 型の I V C 機構においては、アドレスが A 1 から A 2 の範囲であるときは、相手側のデータ RAM 1 5 X にデータを書き込むことによりデータを転送する。したがって、データ RAM の使用効率が高く、また、データの二重持ちを防ぐことができるのでデータに齟齬が発生することも防止できる。また、アドレスが A 3 から A 4 のときは、相手側の P U によりデータが書き込まれたデータ RAM 1 5 X からデータを取得する。したがって、転送されたデータを P U 2 で使用して処理を進めることができる。

## 【 0 0 4 6 】

図 1 2 に、P U T 型の I V C 機構を備えた 4 つの V U P U 1 0 を接続したデータ処理システム 3 0 の例を示してある。この例では、1 つの V U P U 1 0 p が親となり、他の 3 つの V U P U 1 0 c は子供となっており、親の V U P U 1 0 c からは全ての子供の V U P U 1 0 p に対し同様にデータが転送され、子供の V U P U 1 0 c からは親 V U P U 1 0 p へ個別にデータが転送される。このため、親の V U P U 1 0 p は、子供の数に相当する受信 RAM あるいは受信 RAM 領域 1 5 X を備えており、子供の V U P U 1 0 c は 1 つの受信 RAM あるいは受信 RAM 領域 1 5 X を備えている。したがって、親の V U P U 1 0 p においては、子供の V U P U 1 0 c からデータを並列に受信することが可能であり、また、受信した

データを保持しておけるのでプログラムにより適当なときに使用できる。一方、親のVUPU10cの受信RAM15Xを1つにすることも可能であり、この場合は、順番に子供のVUPU10pからデータを受信するように、親のVUPU10pおよび子供のVUPU10cのプログラムを作成する必要がある。

## 【0047】

また、本例のシステムでは、VUPU10pおよびVUPU10cの間では、4本のデータ転送可能な経路を備えたチャンネル35が用意されている。このようなプロセッサ間のデータ転送経路は、一般的な信号通信処理により形成することが可能である。そして、チャンネル数を増加させれば子供のVUPU10c同士が直接通信する構成にすることも可能であり、本発明のIVC機構を備えたVUPUを用いて通信経路を構築するアーキテクチャは自由度が高い。

## 【0048】

図13は、図12に示したデータ処理システム30の各々のVUPUのPUにおけるメモリ構成を示してある。上記と同様に、PUT型のIVC機構を備えたVUPU10を用いているので、1対Nのシステムであってもデータを送出する場合にはますますシステム全体の分散性を高め、かつ、データRAMの使用効率を向上することができる。たとえば、親のVUPU10pのPU(PU-A)においては、メモリマップ19の送信RAMの領域はVUPU10pには実体がなく、そのアドレスに物理的に対応するデータRAMは、子供のVUPU10cにそれぞれ分散して配置されている。また、子供のVUPU10cのPU(PU-B、PU-CおよびPU-D)においても、メモリマップ19の送信RAM領域は実体がなく、それらのアドレスに物理的に対応するデータRAMは、親のVUPU10pに配置されている。

## 【0049】

図14に、本例のIVC機構を実現する通信ユニット12の動作をフローチャートで纏めてある。実際に通信を開始する前に、送信用のコンフィグレーション・レジスタ13Rに、送信先のVUPUのID、送信するデータの開始アドレス（実体のない送信RAMに割り当てられたアドレス）、受信RAM15Xの開始アドレスなどを設定し、受信用のコンフィグレーション・レジスタ14Rに、送

信元となるVUPUのID、送信されるデータの開始アドレス、受信RAMの開始アドレスなどを設定する処理が行われる。これらのコンフィグレーション・レジスタ13Rおよび14Rの設定は、C言語のレベルであればインライン・アセンブル記述により設定できる。また、この処理をファンクションとしてサブルーチン化しておくことも可能である。

#### 【0050】

そして、プログラムにしたがって入出力アドレスが出力されると、通信ユニット12においては、まず、ステップ61でデータの入出力アドレスを判断する。入出力データが通常のデータRAMに割り当てられたアドレスでないときは、ステップ62で、アドレスに基づき出力処理か入力処理かを判断する。入力の場合は、ステップ63で受信RAM15Xに送信されたデータが書き込み中でないこと、すなわち、書き込み状態信号 $\phi put$ の書き込み終了を待ち、ステップ64で自己の受信RAM15Xからデータを読み出す。それと同時に読出し状態信号 $\phi busy$ を読出しにして書き込みを禁止し、読出しが終了すると読出し状態信号 $\phi busy$ を終了状態にする。

#### 【0051】

一方、ステップ62で出力の場合は、ステップ65で読出し状態信号 $\phi busy$ が読出し終了になるのを待ち、ステップ66で出力データ（アドレスとデータおよびそれらアドレスとデータが有効である事を示すライトイネーブル信号）を転送先のVUPU10に送信する。それと共に、書き込み状態信号 $\phi put$ を書き込み状態にして読み出しを禁止し、書き込みが終了すると書き込み状態信号 $\phi put$ を終了状態にする。このように、入出力のアドレスによりデータを通信先のVUPU10のデータRAM15Xにする制御方法を採用することにより、Cレベルの記述でデータの入出力アドレスを管理あるいは制御することだけで複数のVUPU10の間でデータを簡単に交換することができる。

#### 【0052】

図15に、PU-AからPU-Bの受信RAM15Xにデータを書き込む様子をタイミングチャートで示してある。サイクル1では、PU-Bの読出し状態信号 $\phi busy$ がオンになっているので、転送データは有効にならず書き込まれな

い。さらに、通常、読出し状態信号  $\phi b u s y$  がオフとなってから 1 サイクルあけて書込が行われる。このため、サイクル 3 に、P U - A の書込み状態信号  $\phi p u t$  がオンとなり、転送データがアドレス A、データ D およびライトイネーブル W E 込みで受信側の P U - B の受信専用データ R A M 1 5 X に転送される。そして、書込み状態信号  $\phi p u t$  が出力されている間に有効なデータが送信されると、それが受信データ R A M 1 5 X に書き込まれる。この例では 3 サイクルと 5 サイクル目が有効データであることを示している。

## 【 0 0 5 3 】

本発明の I V C 機構においては、図 1 4 に示した処理を通信ユニット 1 2 のファームウェアやゲートロジックで実装することも可能であるが、データ転送のすべてを C レベルの記述で制御することが可能である。図 1 6 ( a ) は、送信側の P U - A の転送手順を C 言語のレベルで記述した例であり、図 1 6 ( b ) は、受信側の P U - B の転送手順を C 言語のレベルで記述した例である。P U - A のプログラム 7 1 では、ステップ 7 1 a でコンフィグレーション・レジスタ 1 3 R に送信スタートアドレスが指定される。ついでステップ 7 1 b で転送相手の受信 R A M にデータを書込むための送信を開始する。この際、ステップ 7 1 c に示すように、送信先の読出し状態信号  $\phi b u s y$  をチェックし、書込み状態信号  $\phi p u t$  をオンにする処理をファンクション・コールによりサブルーチン化しておくことができる。信号のチェックおよび設定が済むと、ステップ 7 1 d で、書き込むためのデータを送出する。そして、データの送出自ら終わるとステップ 7 1 e で終了処理を行うが、ステップ 7 1 f に示すように書込み状態信号  $\phi p u t$  をオフにする処理などをサブルーチン化しておくことができる。

## 【 0 0 5 4 】

一方、P U - B のプログラム 7 2 では、ステップ 7 2 a でコンフィグレーション・レジスタ 1 4 R に受信スタートアドレスが指定される。ステップ 7 2 b で受信 R A M に書込まれた送信元からのデータを読み出す処理を開始する。この際、ステップ 7 2 c に示すように、送信元の書込み状態信号  $\phi p u t$  をチェックし、読出し状態信号  $\phi b u s y$  をオンにする処理をファンクション・コールによりサブルーチン化しておくことができる。信号のチェックおよび設定が済むと、ステ

ップ 7 2 d で、転送されたデータを読み出し、ステップ 7 2 e で読み出し終了処理を行う。ここでも、ステップ 7 2 f に示すように読み出し状態信号  $\phi$  b u s y をオフにする処理などをサブルーチン化しておくことができる。書込み状態信号  $\phi$  p u t および読み出し状態信号  $\phi$  b u s y をオン状態にしたり、その状態を確認するのはレジスタ操作となる。このため、上記のように、ファンクション・コールによりサブルーチン化しておき、別途アセンブラによりレジスタ設定を行う方法が適している。

## 【 0 0 5 5 】

このように、本発明の I V C 機構による通信方法は、データの転送をすべて C 言語のレベルの記述により操作できる。先に説明したように、C 言語による仕様を複数の C 言語によるプロセスに分解して V U P U 化する設計手法により、C 言語による仕様を並列処理および分散処理することができるシステム L S I を設計することが可能であり、この際、データのやりとりが C 言語のレベルで直接記述することにより、V U P U 化するのが容易となる。したがって、本発明の I V C 機構を採用することにより、C 言語による仕様から、並列実行可能な複数の専用回路を備えたシステム L S I を設計および製造する期間を大幅に短縮でき、低コストで提供することができる。

## 【 0 0 5 6 】

図 1 7 は、データを送信する P U - A と、データを受信する P U - B の間の状態情報伝達とそれを構成する信号線を示している。上記の例では、図 1 7 ( a ) に示すように、読取状態信号  $\phi$  b u s y と、書込状態信号  $\phi$  p u t の各々の専用の信号線に情報を持たせている。このため、図 1 7 ( b ) に示すように、それらの状態信号に対応する読取状態提示専用信号線 7 5 と、書込状態提示専用信号線 7 6 が、データを転送する信号線 7 7 に加えて必要になる。

## 【 0 0 5 7 】

これに対し、状態情報の伝達に、受信データ R A M 1 5 X を専用信号線に代わって使用する方法がある。上記の専用信号線を用いた方法では、アセンブラによるレジスタ操作を介して C 言語のレベルから操作する必要があるのに対し、受信データ R A M 1 5 X を使用すると、データに意味を持たせるので、すべて C 言語

のレベルからデータ操作により転送処理を行うことができる。

【 0 0 5 8 】

図 1 8 ( a ) に、送信側の P U - A の転送手順を C 言語のレベルで記述した例を示し、図 1 8 ( b ) は、受信側の P U - B の転送手順を C 言語のレベルで記述した例を示してある。P U - A のプログラム 7 1 では、ステップ 7 1 a でコンフィグレーション・レジスタ 1 3 R に送信スタートアドレスを指定すると共に、ステップ 7 1 g で、自己の受信 R A M 1 5 X のアドレスで、受信側、すなわち、送信先の読取状態信号  $\phi b u s y$  が格納されるアドレスを指定する。送信先の P U - B が受信 R A M 1 5 X を読み出している状態のときは、送信元の受信 R A M 1 5 X の読取状態信号  $\phi b u s y$  が格納されるアドレスにフラグが立つ。したがって、転送相手の受信 R A M にデータを書込むための送信を開始する際は、まず、ステップ 7 1 h で、自己の受信 R A M 1 5 X の読取状態信号  $\phi b u s y$  が格納されるアドレスのデータを参照して送信先の状態をチェックする。ついで、ステップ 7 1 i で、送信先の受信 R A M 1 5 X の受信スタートアドレスにフラグを立てて書き込みを開始したことを伝達する。すなわち、本例では、受信スタートアドレスのデータが書き込み状態信号  $\phi p u t$  が格納されるアドレスとなっている。その後、ステップ 7 1 j で書き込むためのデータを送出し、ステップ 7 1 k で送信先の受信スタートアドレスにフラグをクリアするデータを送出し、書き込みを終了する。

【 0 0 5 9 】

一方、P U - B のプログラム 7 2 では、ステップ 7 2 a でコンフィグレーション・レジスタ 1 4 R に受信スタートアドレスが指定されると共に、ステップ 7 2 g で、送信元の受信 R A M 1 5 X の読取状態信号  $\phi b u s y$  が格納されるアドレスが設定される。受信 R A M 1 5 X に書込まれた送信元からのデータを読み出す処理を開始する際は、まず、ステップ 7 2 h で、書き込み状態信号  $\phi p u t$  が格納される受信スタートアドレスのデータをチェックし、次に、ステップ 7 2 i で、送信元の受信 R A M 1 5 X の読取状態信号  $\phi b u s y$  が格納されるアドレスにデータを送ってフラグを立てる。その後、ステップ 7 2 j で転送されたデータを読取、ステップ 7 2 k で、送信元の受信 R A M 1 5 X の読取状態信号  $\phi b u s y$  が

格納されるアドレスにデータを送ってフラグを解除する。

#### 【 0 0 6 0 】

方式では双方のVUPU10の受信用データRAM15Xに情報を持たせることが前提となるが、VUPU10の間で通信が行われるので、特に制約になることではない。一方、自己の受信用データRAM15Xに、相手方の状態が書き込まれているので、C言語レベルのデータを読み込む処理で相手側が読出状態、あるいは書込み状態の終了を確認できる。

#### 【 0 0 6 1 】

図19は、この方式でデータを送信するPU-Aと、データを受信するPU-Bの間の状態情報伝達とそれを構成する信号線を示している。本例の方式では、読取状態信号 $\phi$ busyと書込状態信号 $\phi$ putの専用の信号線は不要である。したがって、図19(b)に示すように、データを転送する信号線77だけで通信チャンネル35を構成することができ、データを転送するインターフェイスのみで手順の構築が可能となる。しかしながら、その手順はプログラム側にて記載せねばならず、例えば、データ転送の回数をシーケンス番号により表示し、転送漏れが無かったかどうかをプログラム側で判断する、といった操作が必要である。

#### 【 0 0 6 2 】

図20に、本例のVUPUの変形例を示してある。このVUPU10Bは、図8に示した一般のプロセッサ32と通信する機能を備えたVU(COM)を備えているものである。本発明のVUPU10は、上述したIVC機構をVUPU間の通信方式として採用しているものであるが、既に広く使用されているプロセッサには独自のバスプロトコルまたは通信機構を搭載している場合も多く、これら既存プロセッサとVUPU10を通信させることにより、さらにフレキシブルなデータ処理システム30を構築できることは上述した通りである。すなわち、IVC機構により複数のVUPUを用いた分散処理システムを構築したとしても、その中でひとつは既存のプロセッサを使用したいというケースも多い。このような場合にでも本発明にかかるVUPUは有効である。

#### 【 0 0 6 3 】

図20に示したVUPU10BのVU(COM)1Bは、通信ユニット12と



他のCPU 32のバスとのインターフェイスを受け持つバスブリッジ機能26と、通信時のバッファとなるデュアルポートデータRAM 25とを備えている。また、VUPU 10Bにおいては、PU側とVU側との間でレジスタ転送によるVUPUインターフェイスがサポートされているので、PU2からVU1Bへのデータ転送はVUPUインターフェイスを利用できる。したがって、デュアルポートデータRAM 25を他のCPU 32への送信データRAMとして用いることにより、PU2の側から送信を行うことができる。受信はCPU 32のシステムバスと通信ユニット12に受信インターフェイス14をバスブリッジすることにより受信専用データRAM 15Xへ書込むことができる。

## 【0064】

この通信用のVU (COM) 1Bを設けることにより、VUPU 10Bでは、上述したIVC機能では、送信側は相手側のVUPUの受信RAMにデータを書き込むようになっているのに対し、自身の送信データRAM 25にデータを書き込むことになり、実体のある送信データRAMを有するシステムとなる。したがって、IVC機能の多くのメリットのうち、データRAMの利用効率を向上できるメリットは得られない。しかしながら、既存のCPUと本発明による複数のVUPU 10による分散システム30を構築することが可能となり、これらのタイプの異なるプロセッサが共存し、並列に各々の処理を実行可能となるメリットは大きい。

## 【0065】

上記では、本発明にかかるVUPU 10の通信ユニット12がPUT型の場合を例に説明しているが、受信RAM 15Xの代わりに送信RAM 15Yを設けたGET型であっても上記と同様のIVC機能を実現できる。図21にGET型の通信ユニット12を備えたVUPU 10をPU2を中心に示してある。

## 【0066】

GET型の場合は、VUPU 10に送信専用のデータRAM 15Yが設けられており、この送信専用のデータRAM 15Yが通信相手の他のVUPU 10においては受信専用のデータRAMとなる。通信ユニット12も、送信インターフェイス13と受信インターフェイス14とを備えており、各々の制御部13Cおよ

び14Cは、送受信の条件が設定されるコンフィグレーション・レジスタ13Rおよび14Rを備えている。したがって、基本的な構成および動作は上記で説明したPUT型とほぼ同じである。

#### 【0067】

GET型の通信ユニット12の調停回路13Aは、送信専用のデータRAM15Yにデータを書き込む際に、書込み状態信号 $\phi busy$ を書込み状態にして、自分のIDで他のVUPU10に送出して書き込み状態であることを通知する。一方、送信データRAM15Yからのデータの読出しは、通信先の各VUPU10からのリクエスト信号あるいは読出し状態信号 $\phi get$ による。調停回路13Aを有する送信制御部13Cは、リクエスト信号 $\phi get$ が受け入れられ読出し可能な状態となると、受信先のVUPU10のIDを加えた書込み状態信号 $\phi busy$ を読出し可能な状態にして送出し、通信相手のVUPU10へ読出し可能な状態であることを通知する。これにより、通信相手のVUPU10の受信インターフェイス14では、アドレスを送出し、データを読み出す。したがって、PU2が通信先からデータを読み取る場合には、リクエスト信号 $\phi get$ により自分自身へのビジー信号 $\phi busy$ （もちろん、レディ信号 $\phi ready$ であっても良いが）を確認して、受信インターフェイス14に示されるアドレスに応じたデータが受信制御部14Cにより制御されるセクタ16を通じてPU2に供給される。

#### 【0068】

送信専用のデータRAM15Yも上述した受信専用のデータRAM15Xと同様にデュアルポートデータRAMにより構成することが可能である。この場合には送信しながら書込操作が可能となり、並列性が向上する。しかしながら、調停機能を設けない場合は、読出しと書込みのアドレスが同一の場合を考慮して、入力データDIを出力データDOにバイパスする論理回路が必要となる。

#### 【0069】

図22に、GET型のIVC機構を実現する通信ユニット12の動作をフローチャートで纏めてある。実際に通信を開始する前に、送信用のコンフィグレーション・レジスタ13Rに、送信先のVUPUのID、送信RAM15Yの開始ア

ドレス、受信するデータの開始アドレス（実体のない受信 RAM に割り当てられたアドレス）などを設定し、受信用のコンフィグレーション・レジスタ 1 4 R に、受信元となる VUPU の ID、送信 RAM の開始アドレス、受信されるデータの開始アドレス、などを設定する処理が行われる。これらのコンフィグレーション・レジスタ 1 3 R および 1 4 R の設定は、C 言語のレベルであればインライン・アセンブル記述により設定できる。また、この処理をファンクションとしてサブルーチン化しておくことができる。

## 【 0 0 7 0 】

そして、プログラムにしたがって入出力アドレスが出力されると、通信ユニット 1 2 においては、まず、ステップ 8 1 でデータの入出力アドレスを判断する。入出力データが通常のデータ RAM に割り当てられたアドレスでないときは、ステップ 8 2 で、アドレスに基づき出力処理か入力処理かを判断する。出力の場合は、ステップ 8 3 で送信 RAM 1 5 Y が読出し中でないこと、すなわち、読出し状態信号（リクエスト信号） $\phi get$  の読出し終了を待ち、ステップ 8 4 で自己の送信 RAM 1 5 Y にデータを書き込む。それと同時に書込み状態信号  $\phi busy$  を書込みにして読出しを禁止し、書込みが終了すると状態信号  $\phi busy$  を終了状態にする。

## 【 0 0 7 1 】

一方、ステップ 8 2 で入力の場合は、リクエスト信号  $\phi get$  を送出し、ステップ 8 5 で状態信号  $\phi busy$  が書込み終了になるのを待ち、ステップ 8 6 でデータを通信先の VUPU 1 0 から受領する。そして、読出しが終了すると、リクエスト信号  $\phi get$  を終了状態にする。このように、GET 型においても、入出力のアドレスによりデータを受信先の VUPU 1 0 のデータ RAM 1 5 Y から取得する制御方法を採用することにより、C レベルの記述でデータの入出力アドレスを管理あるいは制御することだけで複数の VUPU 1 0 の間でデータを簡単に交換することができる。そして、このような処理を通信ユニット 1 2 のファームウェアやゲートロジックで行っても良く、あるいは C 言語のレベルで記述することも可能である。

## 【 0 0 7 2 】

上述したPUT型の通信方法とGET型の通信方法は、どちらもC言語から直接データをアクセスできる点では同じであり、自己のVUPUのデータRAMにアクセスするのと同じ操作で他のVUPUのデータRAMにデータを書込・読出することによりデータ交換を行うことができる。PUT型の通信方法のVUPU10を採用したデータ処理システム30では、親のVUPU10pあるいは他のプロセッサが、共通のデータを複数の子のVUPU10cに転送し、子のVUPU10cは転送されたデータを頻繁にアクセスし、かつ加工し、処理を進める分散処理に適している。一方、GET型の通信方法のVUPU10を採用したデータ処理システム30は、親のVUPU10pあるいは他のプロセッサから子のVUPU10cに供給されるデータが少量であり、さらに、子のVUPU10cがデータをそれぞれ独立に参照しながら処理を進める分散処理に適している。

## 【0073】

さらに、目的に応じてPUT型とGET型を両立させたデータ処理システムを構築することも可能である。例えば、親のVUPU10pのデータを少量ずつ複数の子のVUPU10cが参照しながら各々分散処理を行い、その結果を親のVUPU10pに戻す処理が要求される場合がある。このような処理に対応したデータ処理システム30としては、GET型の通信方法で親のVUPU10pから子のVUPU10cにデータを転送し、PUT型の通信方法で子のVUPU10cから親のVUPU10pにデータを返却する方式が最もメモリ効率がよい。なぜなら、送信専用・受信専用データRAMを親のVUPU10pがひとつ持てばよいからである。また、ひとつの親のVUPU10pと複数の子のVUPU10cにより分散処理を行う構造のデータ処理システム30は、本発明にかかるVUPU10を用いた極めて基本的な構造であると考えられる。したがって、親のVUPU10pにのみ転送専用メモリを保有させて共有化を図るデータ処理システムは、本発明のVUPU10を用いた有効な分散処理の基本構造であるといえる。

## 【0074】

図23に、送信専用のデータRAM15Yと受信専用のデータRAM15Xを有するVUPU10pの構成例を示してある。このVUPU10pにおいては、

通信ユニット 1 2 の送信インターフェイス 1 3 は上述した G E T 型の構成であり、送信専用のデータ R A M 1 5 Y を制御し、子の V U P U 1 0 c のそれぞれからのリクエスト信号  $\phi$  g e t に基づいてデータ転送を行う。受信インターフェイス 1 4 は、P U T 型の構成であり、子の V U P U 1 0 c のそれぞれからの書込要求信号  $\phi$  p u t に基づいてデータの書込を行う。

## 【 0 0 7 5 】

なお、以上では、通常のデータ R A M 1 5 N、受信専用のデータ R A M 1 5 X および送信専用のデータ R A M 1 5 Y がそれぞれ独立している構成を例に説明しているが、同一のデータ R A M の領域を割り振ることで対応することも可能である。しかしながら、受信専用および送信専用のデータ R A M はデュアルポート R A M あるいは多ポート R A M を採用することによるメリットがあり、通信容量が小さくて良いデータ処理システムにおいては、受信専用あるいは送信専用のデータ R A M を独立して設けることが望ましい。

## 【 0 0 7 6 】

## 【発明の効果】

以上に説明したように、本発明においては、専用データ処理ユニット (V U) と汎用データ処理ユニット (P U) とを有するデータ処理装置 (V U P U) において、P U に通信機能を持たせることにより、複数の V U、すなわち専用回路を並列に実行できるデータ処理システムを極めて短期間に、そして低コストで開発することができる。システム L S I として与えられた仕様全体をハードウェア化する作業は膨大であり時間と経費の点から現在ではほとんど経済的に見合わないものとなっている。これに対し、本発明の V U P U は、システム L S I として与えられた仕様の内、ハードウェア化するのに適した機能を適当な単位で抽出し、シミュレーションによって高速化などの効果が確認された機能だけを V U としてハードウェア化することができる。したがって、ハードウェア化される範囲は限られたものとなり、容易に設計および開発でき、また費用も最小限で済む。その一方で、ハードウェア化したことによる効果は最大限にすることが可能である。それに加えて、ハードウェア化した V U を並列に実行することができるので、処理を複数の V U に分散することが可能となり、処理効率が高く処理速度の速い、

さらに経済的なデータ処理システムを提供することが可能となる。

【 0 0 7 7 】

さらに、本発明のVUPUは、繰り返し計算の多い処理などを機能単位で抽出してVUとして実現し高速処理を可能とすると共に、他の処理は汎用プロセッサであるPUで処理することによって、ハードウェア化に伴うコストアップや設計期間の長期化を抑制し、さらに、仕様変更や、開発のあらゆる段階の変更にも柔軟に対処できるというメリットも備えている。そして、プログラムレベルで制御できるPUに通信機能を設けることにより、プログラムレベルで並列処理の制御を行うことが可能となり、極めて柔軟な制御が可能となる。その結果、高級言語で記述された仕様に基づくシステムLSIを極めて短い期間で設計および開発できる。

【 0 0 7 8 】

このようなVUPUを用いてデータ処理システムを構築することにより、ひとつのC言語などの高級言語で記述されたプロセスを分割して複数の処理プロセスにし、これらプロセス間のデータの転送と処理依頼、それに基づく処理結果の返却を設計するにはデータ転送に関してC言語あるいはJAVA言語などの高級言語との親和性が高く、かつハードウェアを意識しないでデータ転送の設計が進められる方式が不可欠となる。上述した本発明によれば、アドレスによって、通信先のVUPUの受信専用のデータRAMにデータを送信し、あるいは通信先のVUPUの送信専用のデータRAMからデータを取得することができる。このため、VUPU間の通信を、メモリへのアクセスと同じ方法にてC言語などから直接に行うことができ、極めて自由にデータ送受を行うことができる。このため、複数のC言語により記述されたプロセスが並列に動作するように設計することが極めて容易となる。

【 0 0 7 9 】

このように、C言語あるいはその他の高級言語との連動性および対応性のあるデータ通信機構をハードウェア側に設けることにより、極めて容易に、高級言語のレベルでデータ転送が記述できる。その結果、高級言語により記述されたプロセスを複数に分割が容易となり、分散処理システムの設計が可能となる。したが

って、本発明で開示した通信機構は、上述した複数のVUPUを用いた処理速度の速いデータ処理システムを構築するのに好適なものである。

【図面の簡単な説明】

【図 1】

本発明に係るPUおよびVUを備えたデータ処理装置（VUPU）の概要を示す図である。

【図 2】

C言語により記述されたプロセスを複数に分解する様子を示す図である。

【図 3】

本発明のデータ処理装置により分散処理するデータ処理システムを構築する例を示す図である。

【図 4】

図 3 に示したデータ処理システムの各VUPUの実行状態を例示する図である。

【図 5】

C言語で記述したプログラムが分散処理用に分解される様子を示す図である。

【図 6】

本発明のデータ処理装置により分散処理するデータ処理システムの異なる例を示す図である。

【図 7】

本発明のデータ処理装置により分散処理するデータ処理システムのさらに異なる例を示す図である。

【図 8】

本発明のデータ処理装置により分散処理するデータ処理システムのさらに異なる例を示す図である。

【図 9】

C言語で記述された機能をVUPU化する過程の概要を示す図である。

【図 1 0】

本発明の通信機能を有するVUPUの概略構成をPUの構成を中心に示す図で

ある。

【図 1 1】

2つのVUPUで交信する際のメモリの使用状況を示す図である。

【図 1 2】

親のVUPUと複数の子のVUPUで交信するデータ処理システムの概要を示す図である。

【図 1 3】

図 1 2 に示すデータ処理システムの各PUのメモリマップを示す図である。

【図 1 4】

通信ユニットの処理の概要を示すフローチャートである。

【図 1 5】

受信RAMにデータを入出力するタイミングを示す図である。

【図 1 6】

通信ユニットの処理をC言語により制御するプログラム例を示す図である。

【図 1 7】

調停を行うための状態信号とそれに対応する信号線を示す図である。

【図 1 8】

状態信号を受信RAMに書き込む通信方式の処理をC言語により制御するプログラム例を示す図である。

【図 1 9】

状態信号を受信RAMに書き込む通信方式における状態信号と信号線とを示す図である。

【図 2 0】

本発明の通信機能を有するVUPUの概略構成をPUの構成を中心に示す図であり、他のCPUとの通信機能を備えたVU(COM)を有するVUPUの概要を示す図である。

【図 2 1】

本発明の通信機能を有するVUPUの概略構成をPUの構成を中心に示す図であり、GET型の通信機能を有するVUPUを示す図である。



【図 2 2】

図 2 1 に示す V U P U の通信ユニットの処理の概要を示すフローチャートである。

【図 2 3】

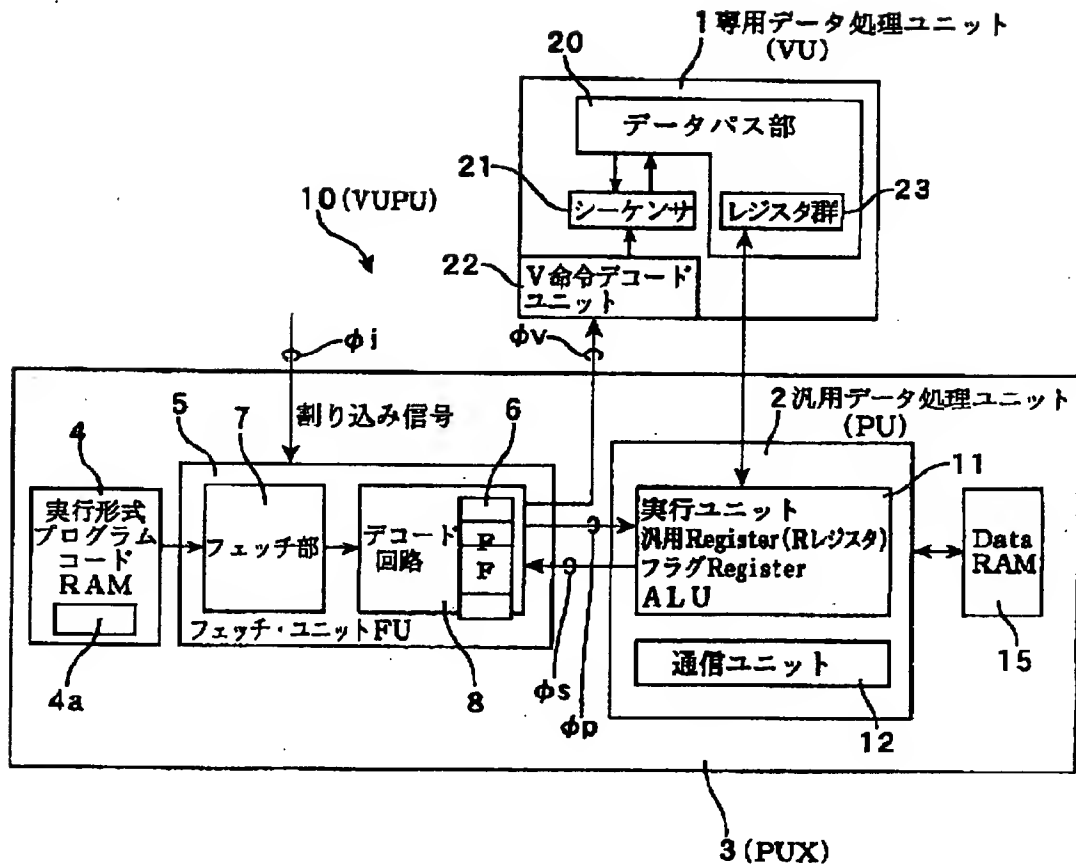
本発明の通信機能を有する V U P U の概略構成を P U の構成を中心に示す図であり、 P U T - G E T 型の通信機能を有する V U P U を示す図である。

【符号の説明】

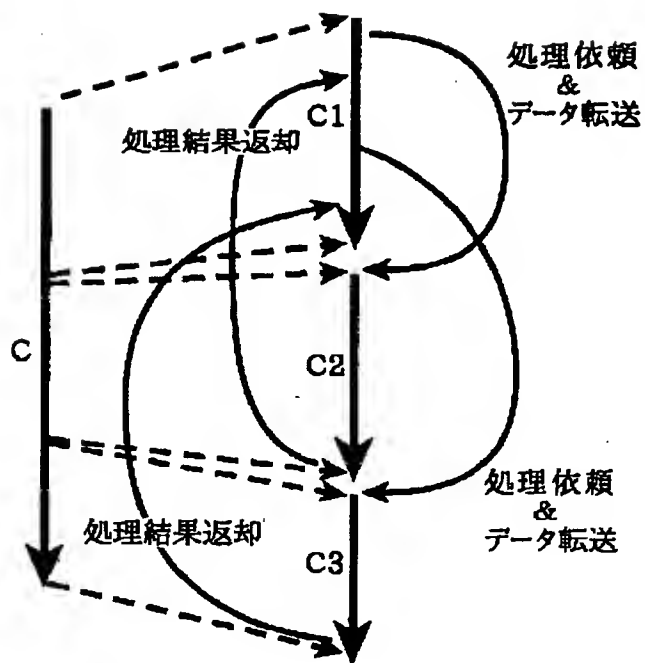
- 1     専用データ処理ユニット V U
- 2     汎用データ処理ユニット P U
- 3     汎用プロセッサ P U X
- 4     コード R A M
- 4 a    制御プログラム
- 5     フェッチユニット F U
- 1 0    データ処理装置 ( V U P U )
- 1 1    実行ユニット
- 1 2    通信ユニット
- 1 3    送信インターフェイス、 1 4    受信インターフェイス
- 1 5 N    R D / W R データ R A M
- 1 5 X    受信専用のデータ R A M、 1 5 Y    送信専用のデータ R A M

【書類名】 図面

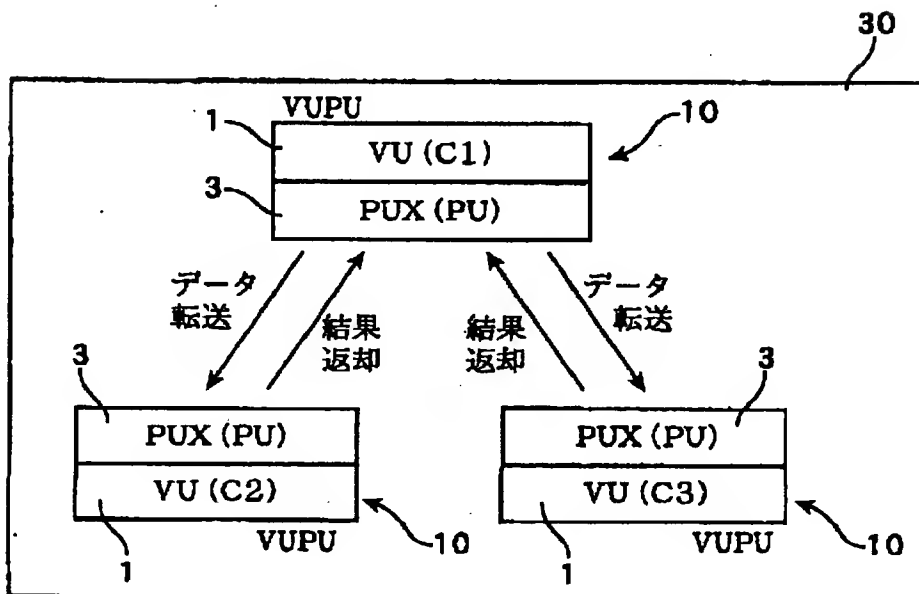
【図 1】



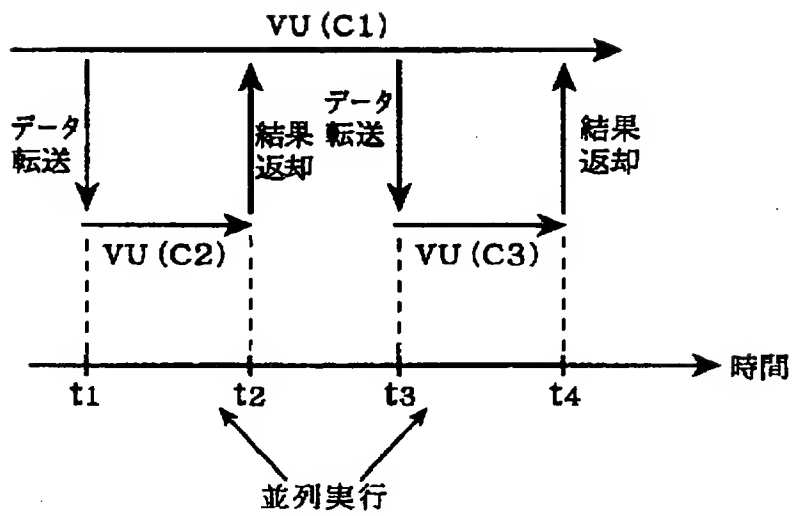
【図 2】



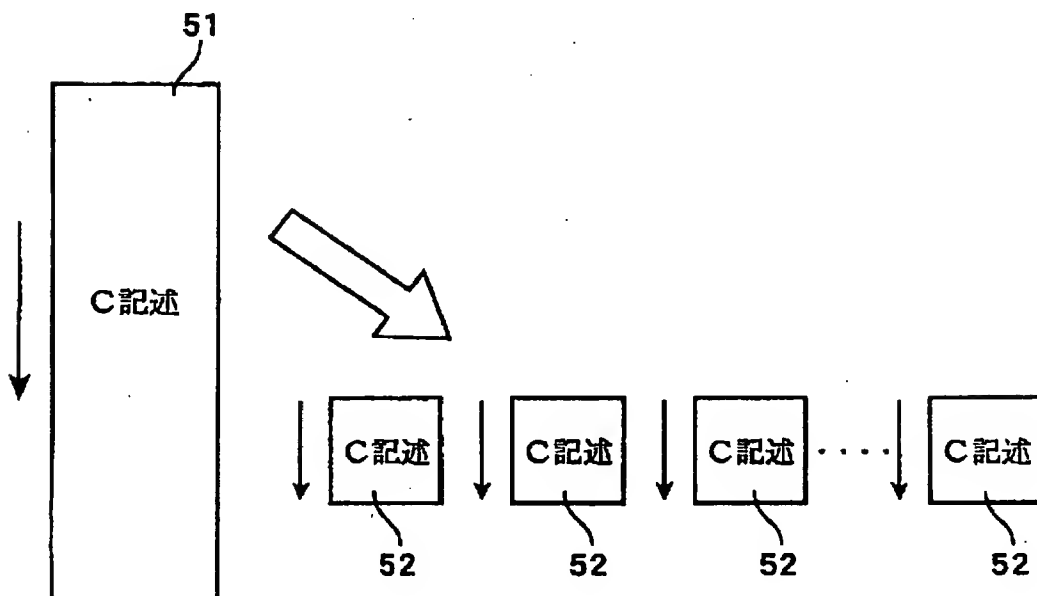
【図 3】



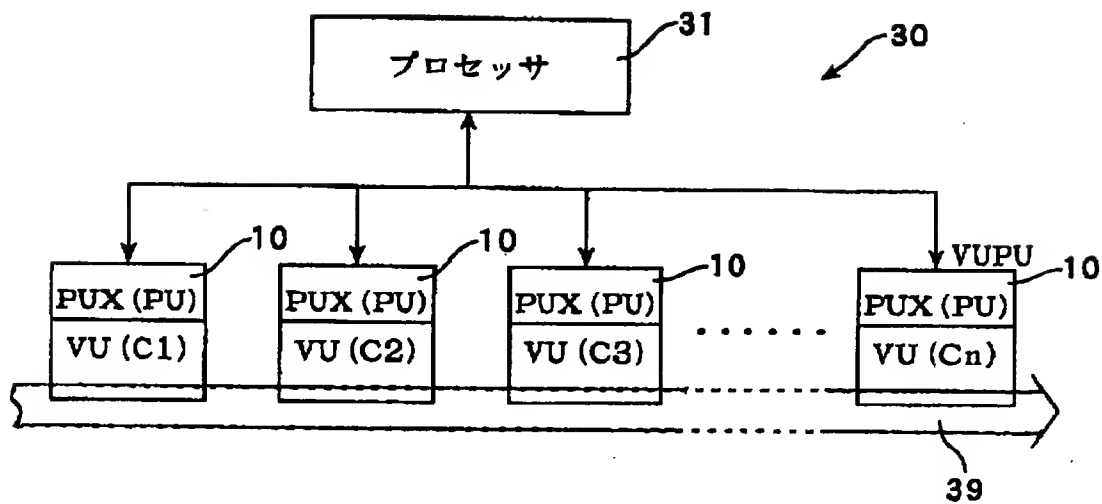
【図 4】



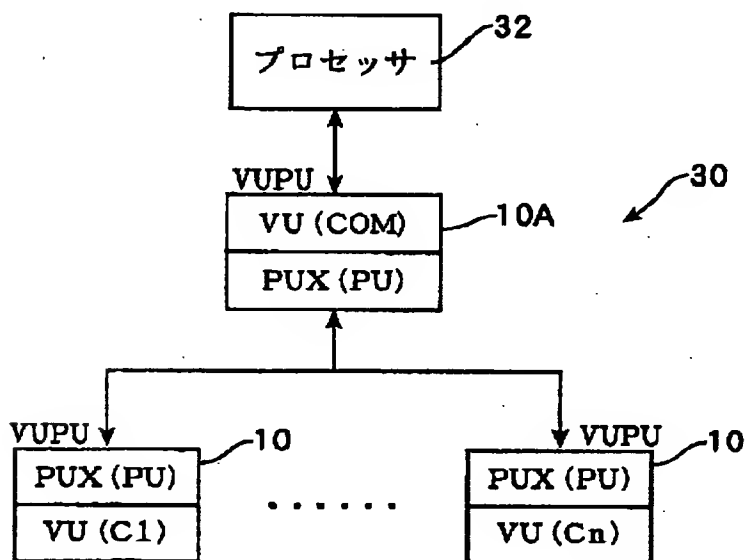
【図 5】



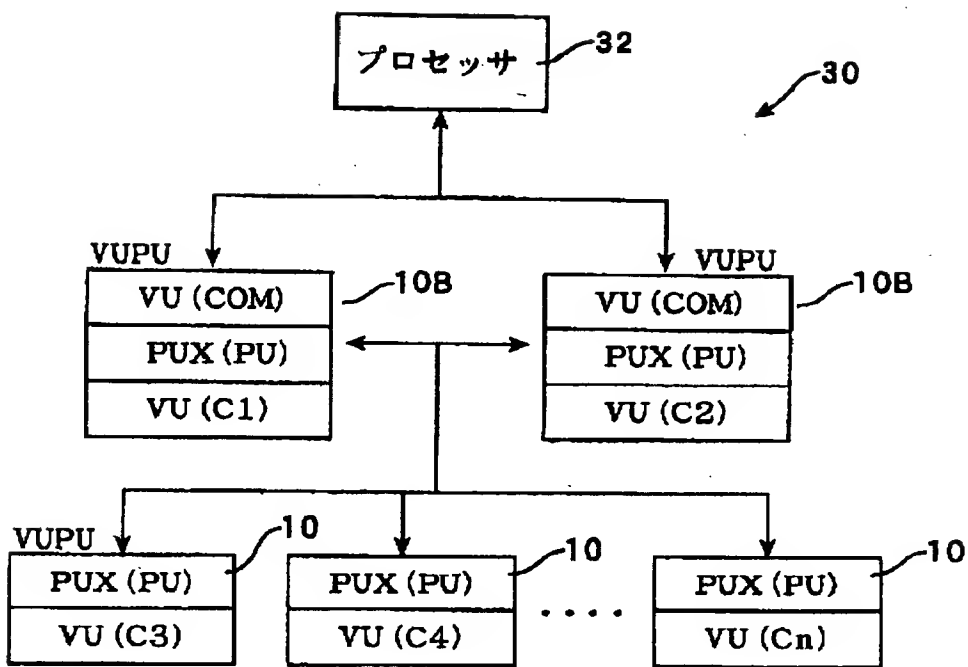
【図 6】



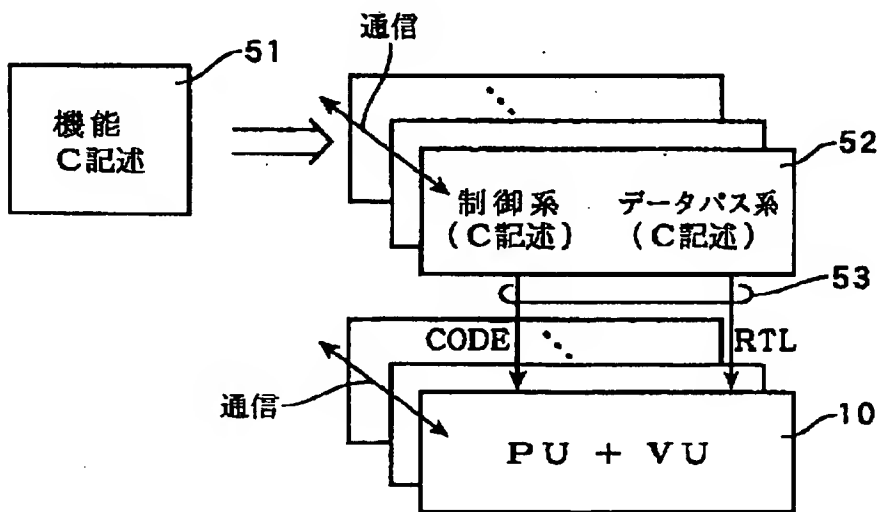
【図 7】



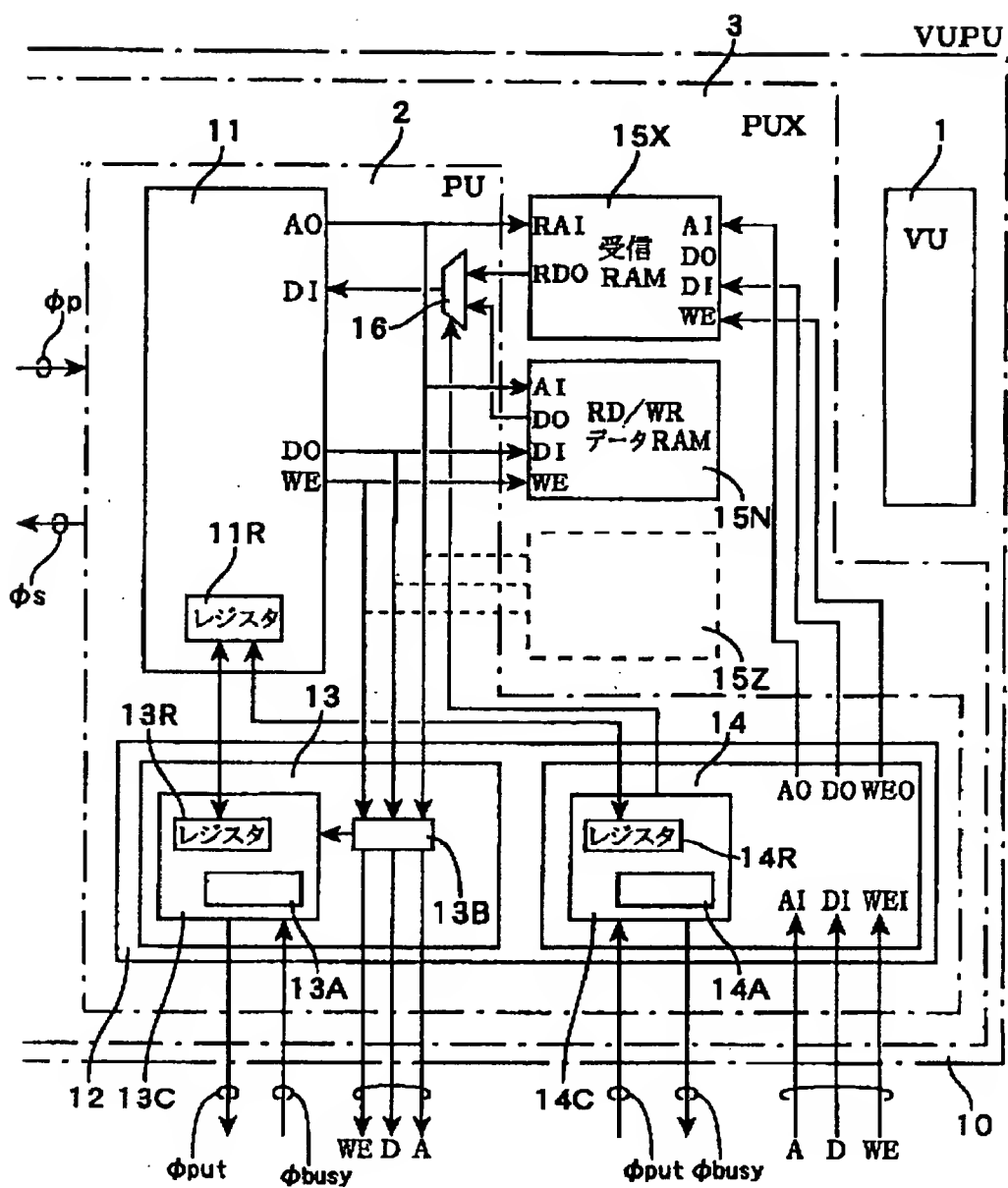
【図 8】



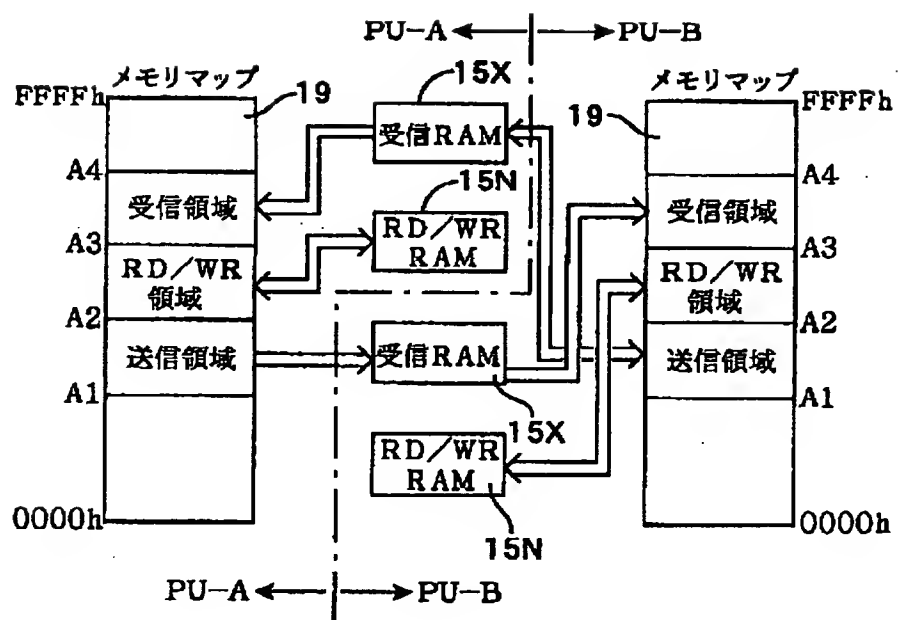
【図 9】



【図 10】

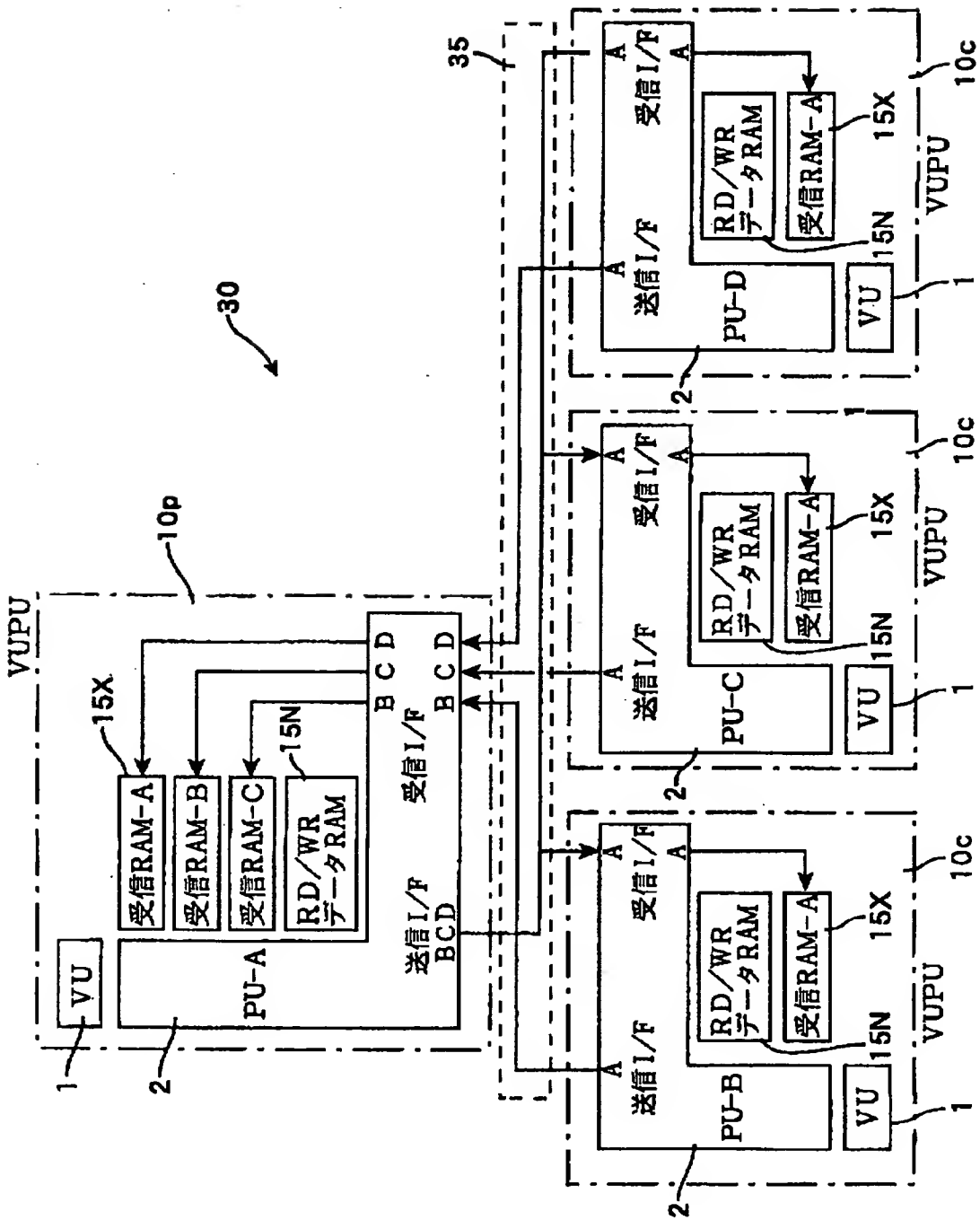


【図 1 1】

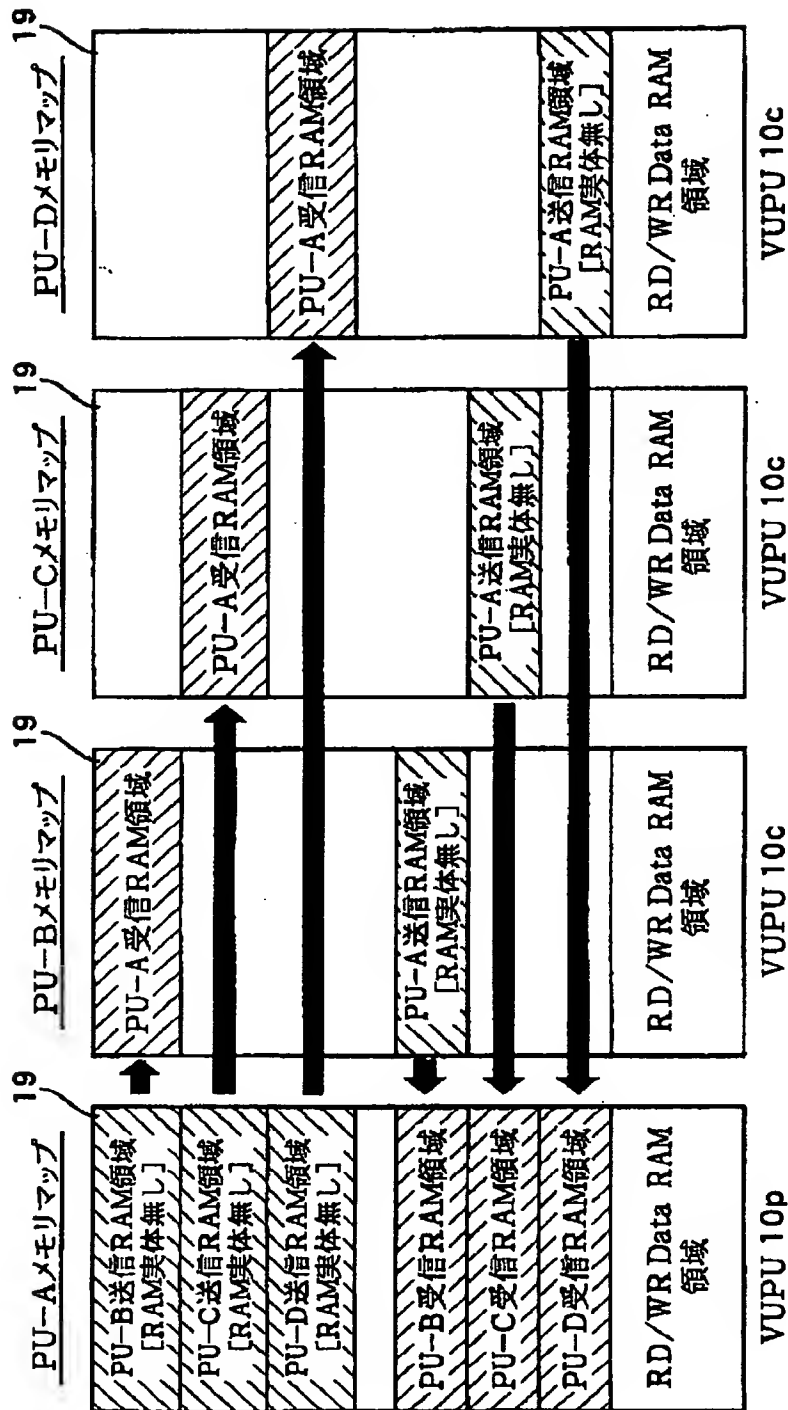




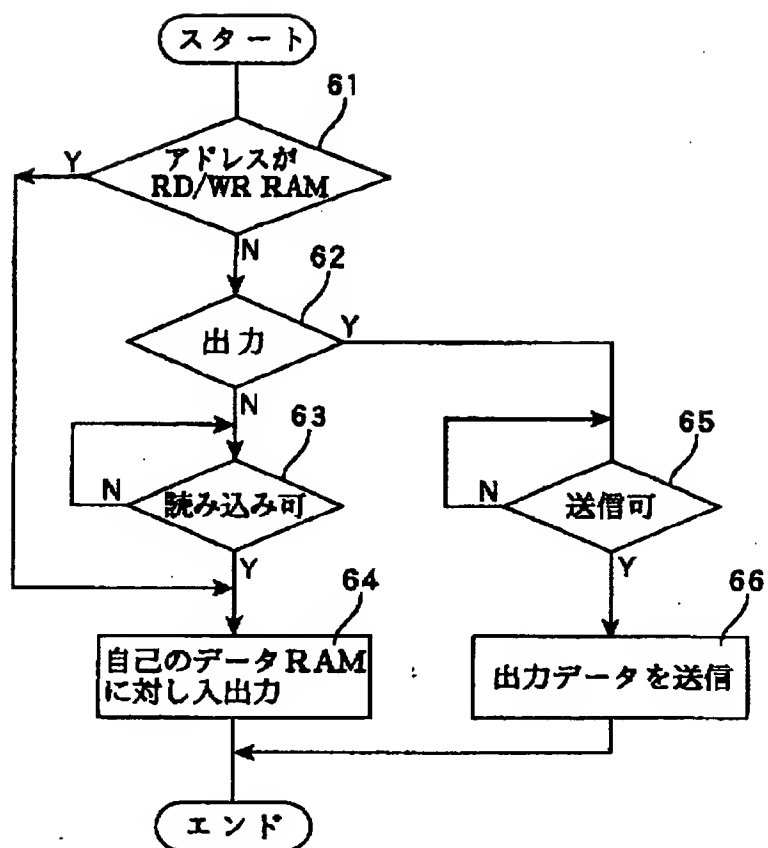
【図12】



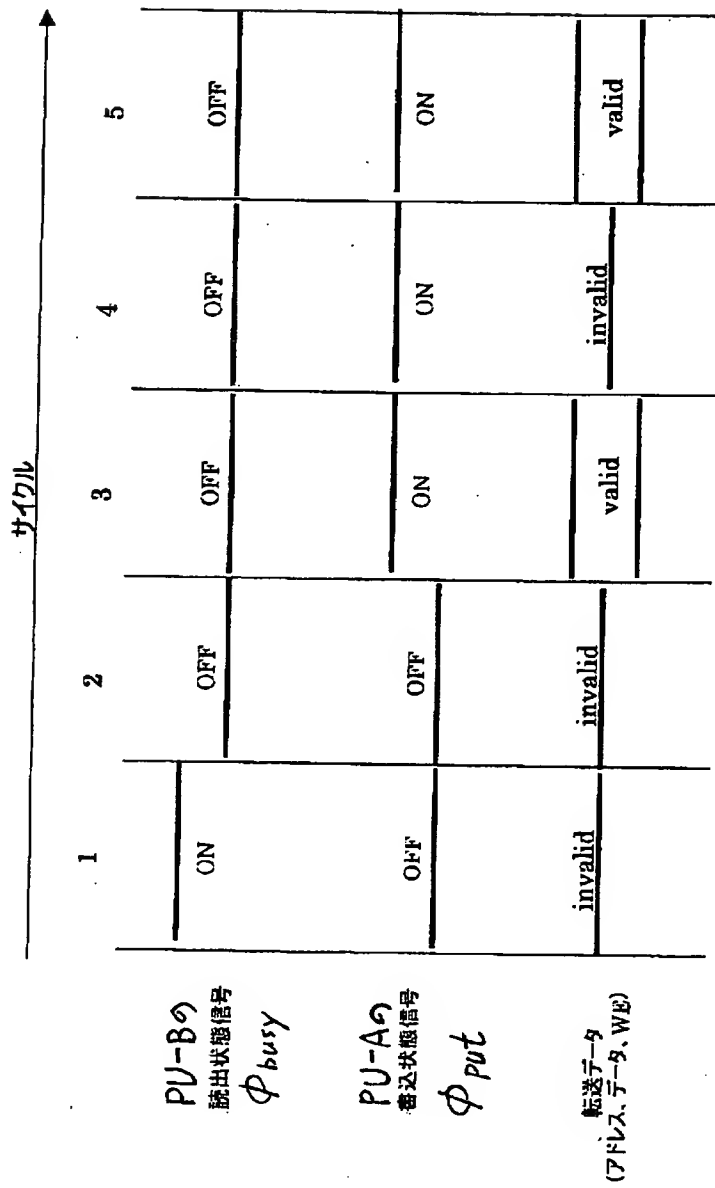
【図 13】



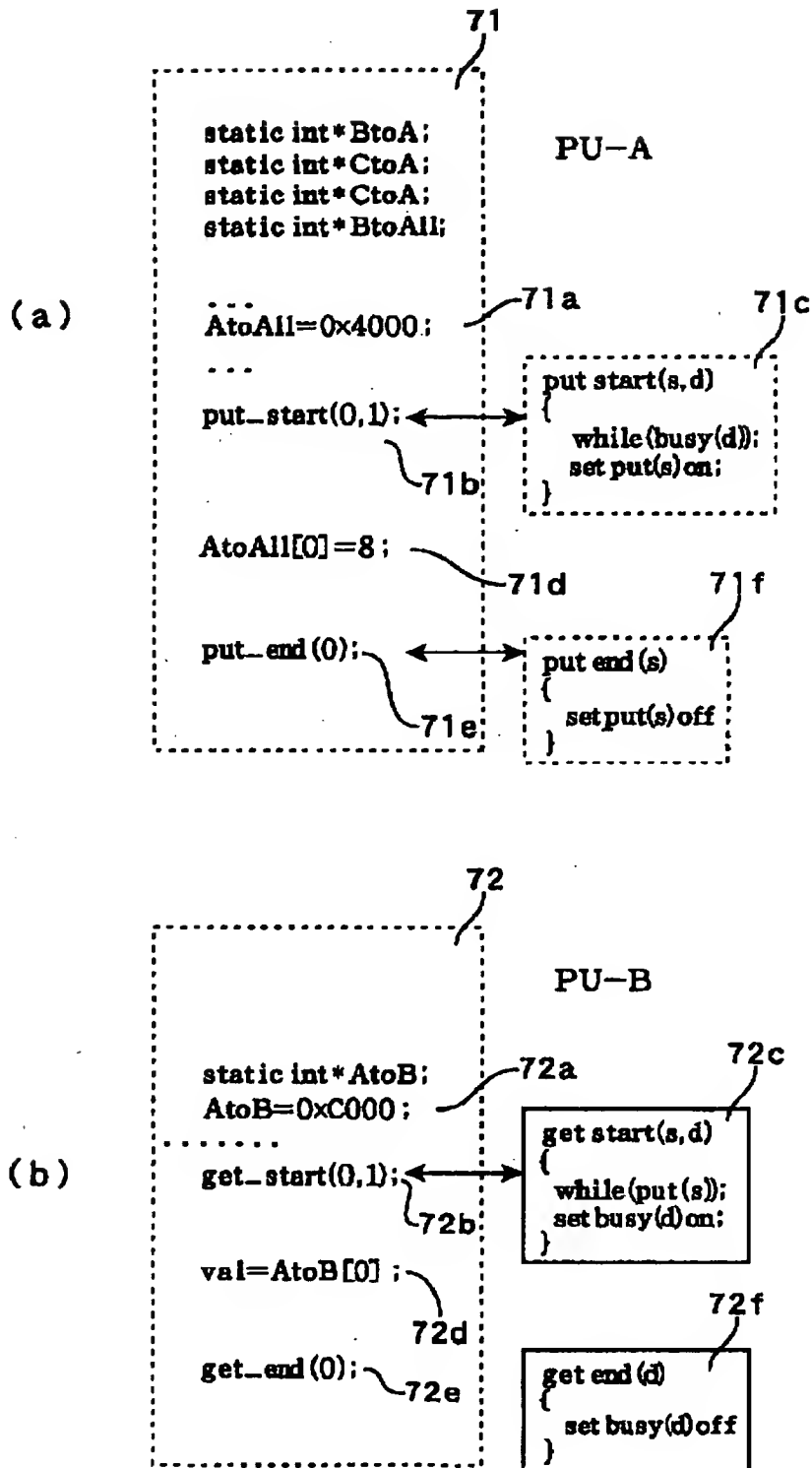
【図 1 4】



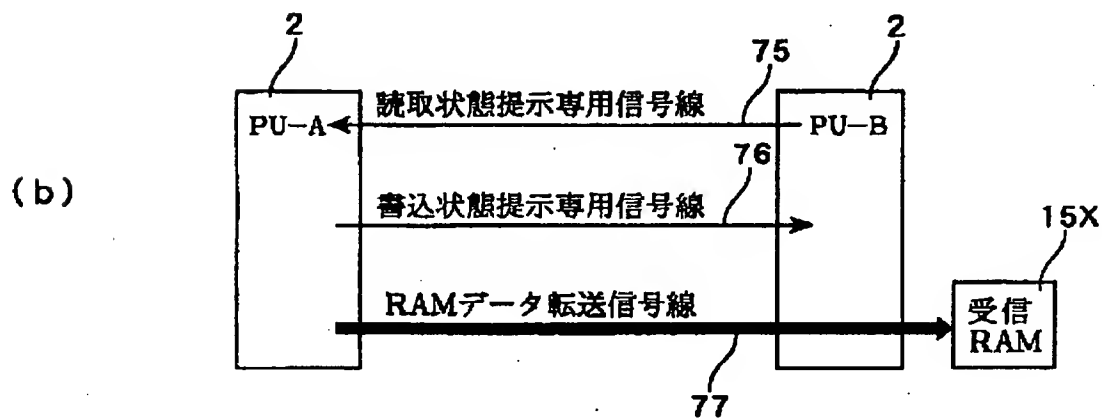
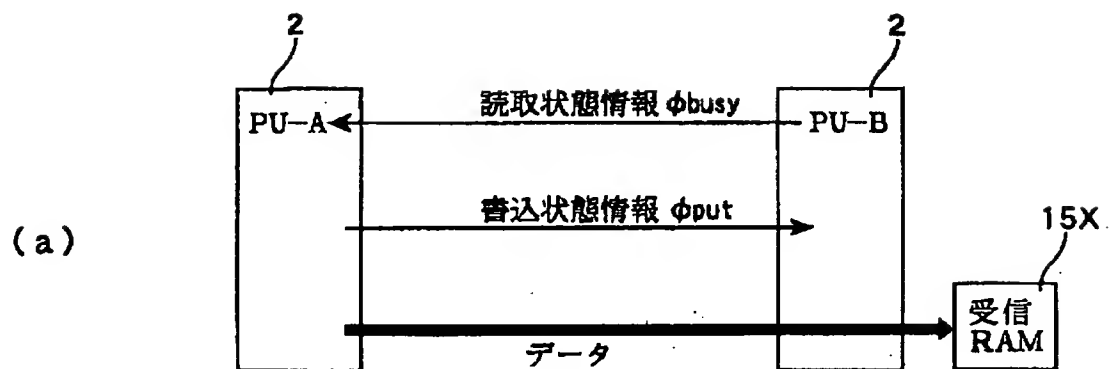
【図 15】



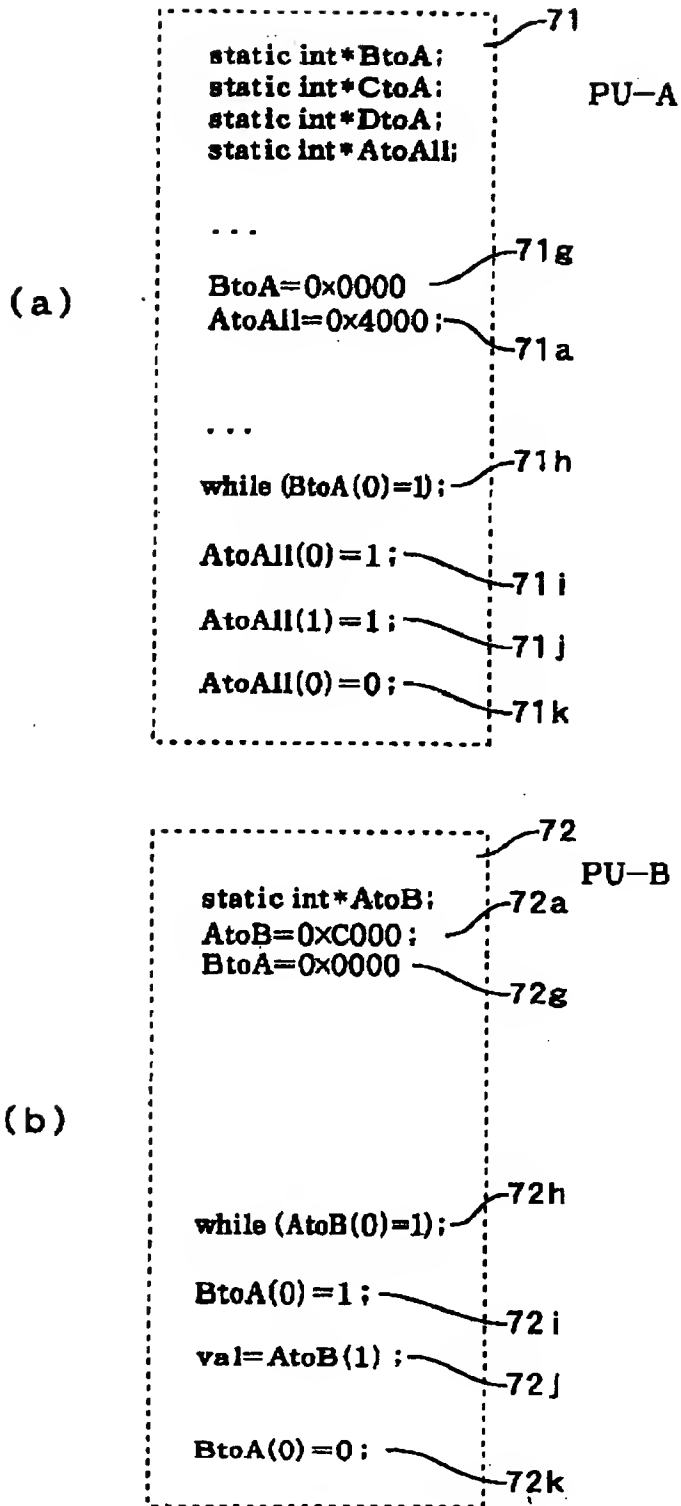
【図 16】



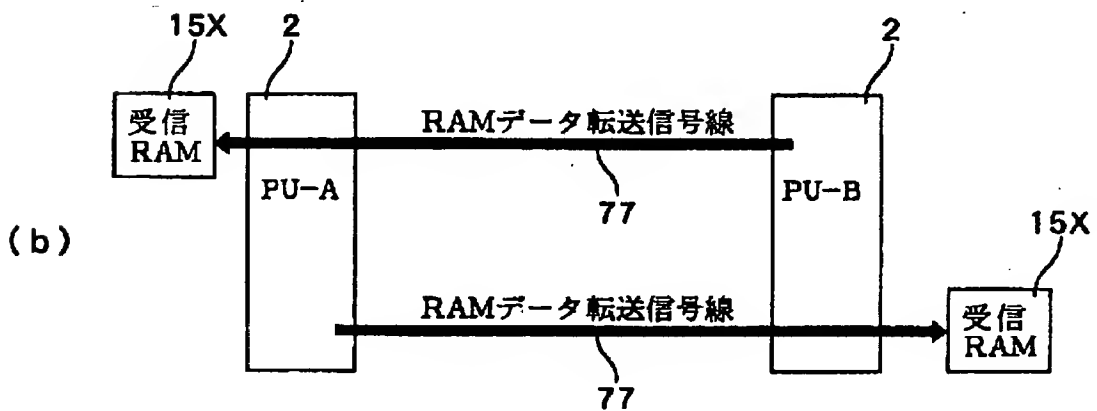
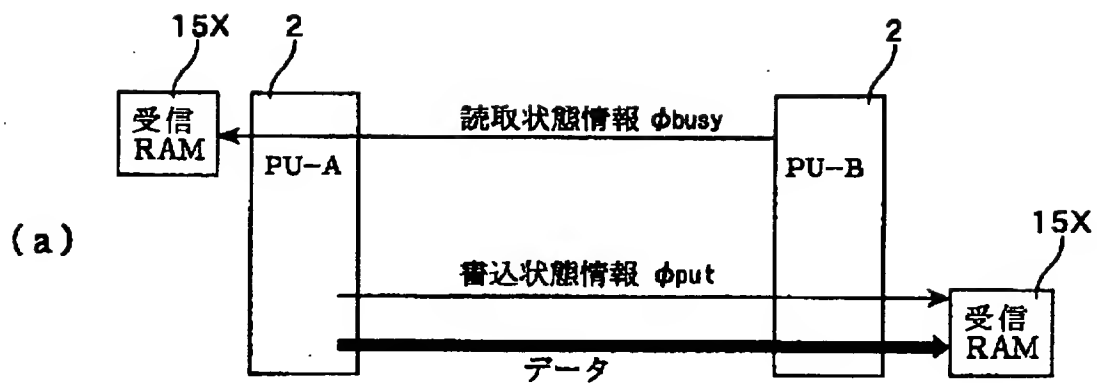
【図 1 7】



【図 1 8】

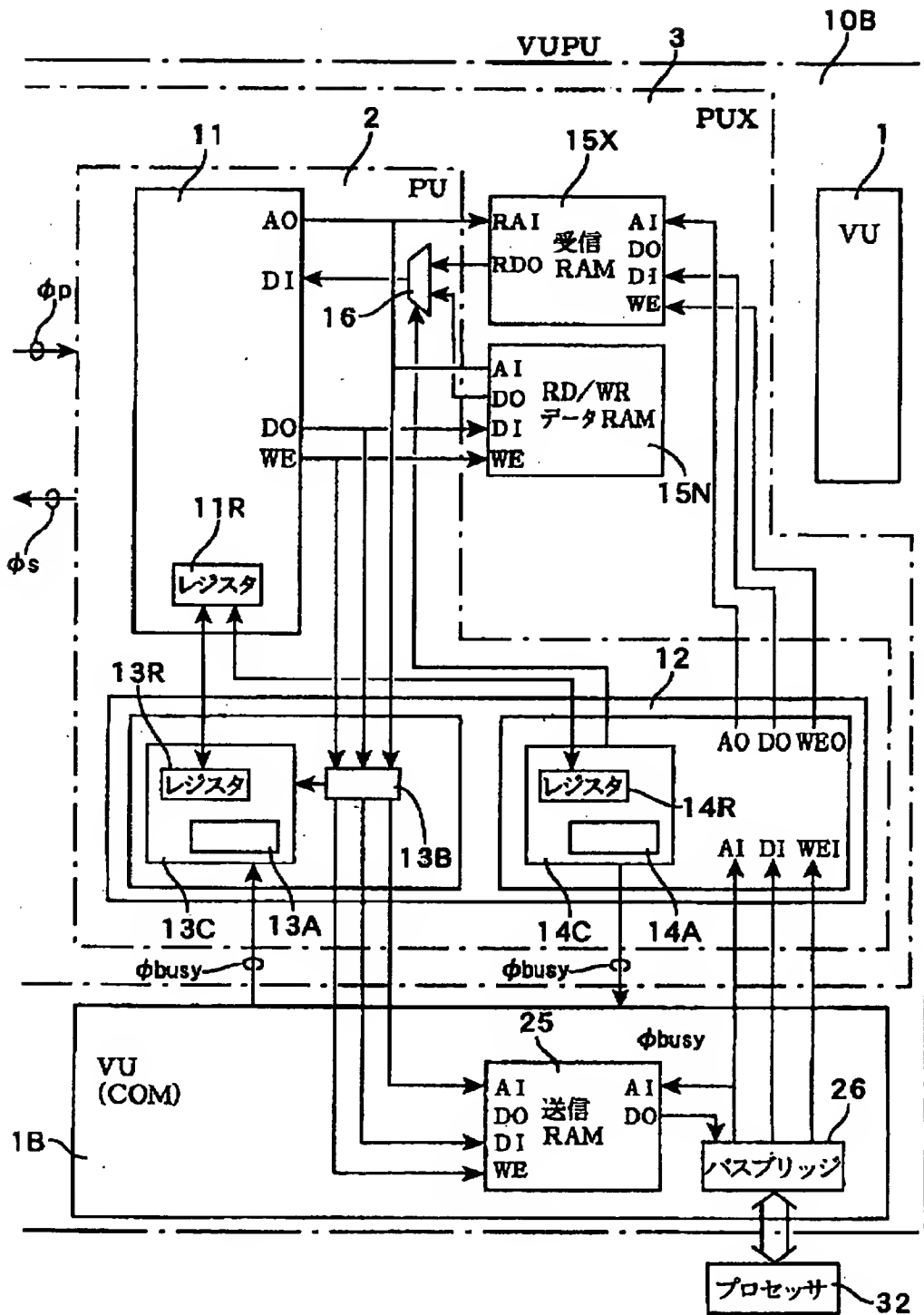


【図 1 9】



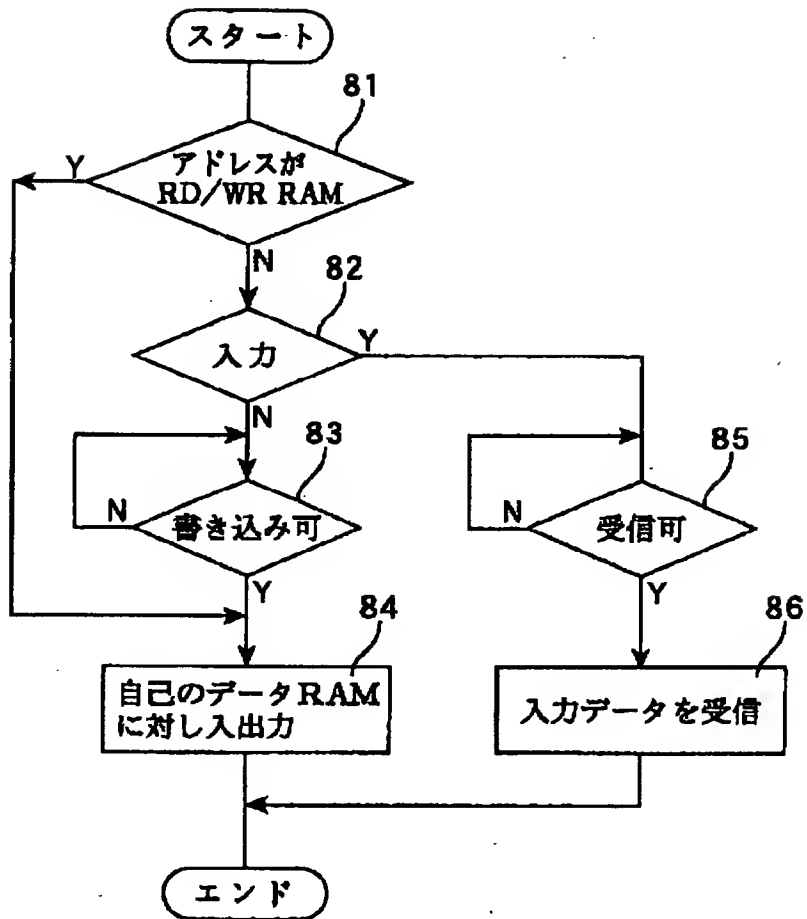


【図 20】

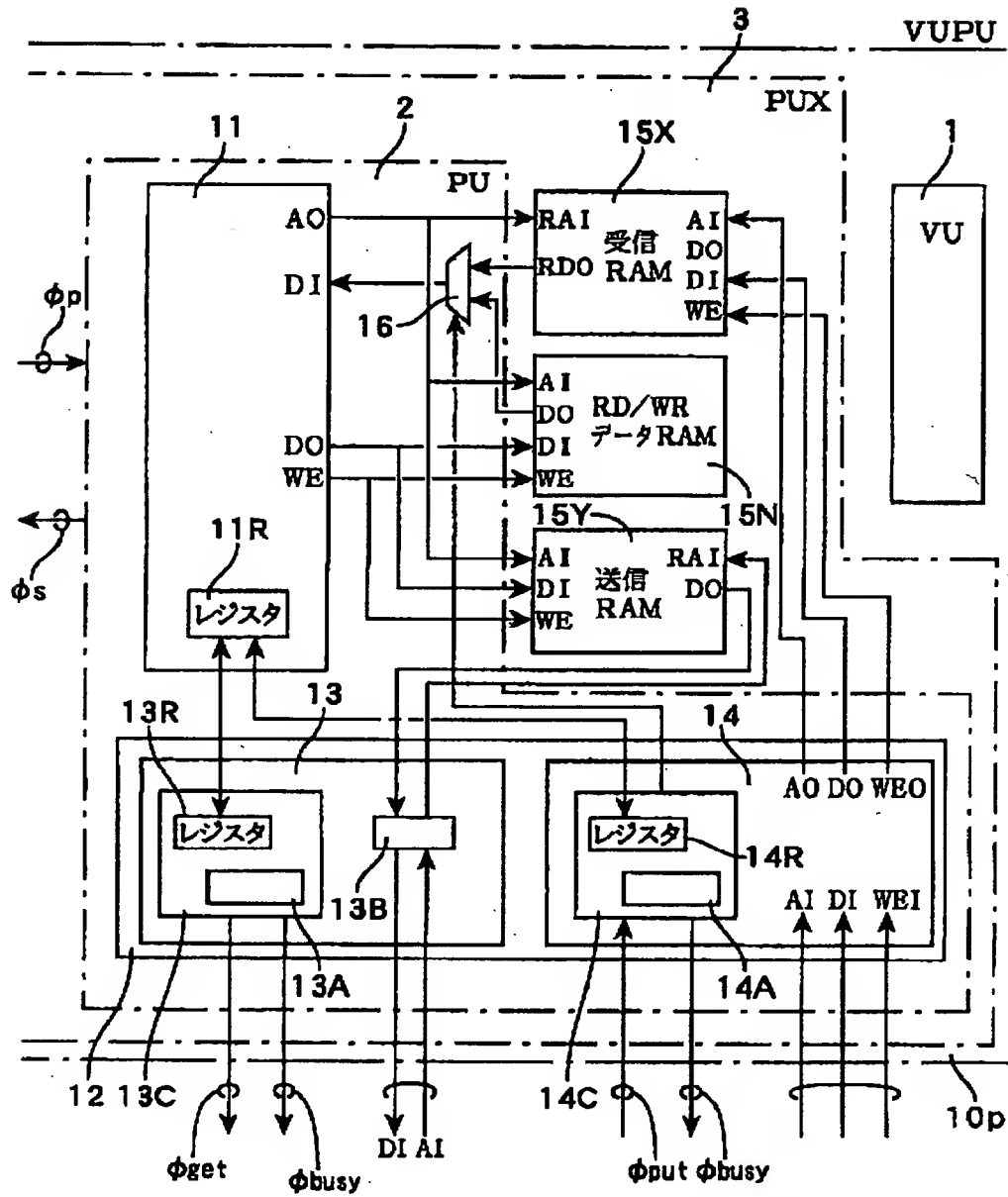




【図 22】



【図 23】



【書類名】 要約書

【要約】

【課題】 C言語などの高級言語で与えられる仕様を高速で処理可能なデータ処理システムを短期間で経済的に構築可能とする。

【解決手段】 汎用処理が可能な汎用データ処理ユニットPU2と、特定のデータ処理をハードウェア化して高速で実行可能な専用データ処理ユニットVU1とを有するデータ処理装置10において、汎用データ処理ユニットPU2に通信ユニット12を設けることにより、複数の専用データ処理ユニットVU1が並列に稼動できるデータ処理システムを構築可能とする。

【選択図】 図10

認定・付加情報

特許出願の番号	特願 2001-024513
受付番号	50100137714
書類名	特許願
担当官	第七担当上席 0096
作成日	平成13年 2月 1日

<認定情報・付加情報>

【提出日】	平成13年 1月31日
-------	-------------

出 願 人 履 歴 情 報

識別番号 [598149242]

1. 変更年月日 1998年10月29日  
[変更理由] 新規登録  
住 所 東京都新宿区西新宿6丁目12番1号  
氏 名 パシフィック・デザイン株式会社